Package ‘rstiefel’

October 14, 2022

Type Package

Title Random Orthonormal Matrix Generation and Optimization on the Stiefel Manifold

Version 1.0.1

Date 2021-06-14

Author Peter Hoff and Alexander Franks

Maintainer Peter Hoff <peter.hoff@duke.edu>

Description Simulation of random orthonormal matrices from linear and quadratic exponential family distributions on the Stiefel manifold. The most general type of distribution covered is the matrix-variate Bingham-von Mises-Fisher distribution. Most of the simulation methods are presented in Hoff(2009) ``Simulation of the Matrix Bingham-von Mises-Fisher Distribution, With Applications to Multivariate and Relational Data'' <doi:10.1198/jcgs.2009.07177>. The package also includes functions for optimization on the Stiefel manifold based on algorithms described in Wen and Yin (2013) ``A feasible method for optimization with orthogonality constraints'' <doi:10.1007/s10107-012-0584-1>.

License GPL-3

RoxygenNote 6.0.1

Depends R (>= 2.10)

Suggests knitr

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-06-15 15:40:02 UTC

R topics documented:

rstiefel-package 2
lineSearch 3
lineSearchBB 4
NullC 5
optStiefel 6

rbing.matrix.gibbs 7
rbing.O2 9
rbing.Op 10
rbing.vector.gibbs 11
rbmf.matrix.gibbs 13
rbmf.O2 14
rbmf.vector.gibbs 15
rmf.matrix 16
rmf.matrix.gibbs 18
rmf.vector 19
rustiefel 20
rW 21
ry_bing 22
ry_bmf 23
tr 24

Index **25**

`rstiefel-package` *Random Orthonormal Matrix Generation on the Stiefel Manifold #' Simulation of random orthonormal matrices from linear and quadratic exponential family distributions on the Stiefel manifold. The most general type of distribution covered is the matrix-variate Bingham-von Mises-Fisher distribution. Most of the simulation methods are presented in Hoff(2009) "Simulation of the Matrix Bingham-von Mises-Fisher Distribution, With Applications to Multivariate and Relational Data" <doi:10.1198/jcgs.2009.07177>. The package also includes functions for optimzation on the Stiefel manifold based on algoirthms described in Wen and Yin (2013) "A feasible method for optimization with orthogonality constraints" <doi:10.1007/s10107-012-0584-1>.*

Description

Package:	rstiefel
Type:	Package
Version:	1.0.0
Date:	2019-02-18
License:	GPL-3

Author(s)

Peter Hoff

Alex Franks

Maintainer: Peter Hoff <peter.hoff@duke.edu>

References

Hoff(2009)

Examples

```
Z<-matrix(rnorm(10*5),10,5) ; A<-t(Z)%*%Z
B<-diag(sort(rexp(5),decreasing=TRUE))
U<-rbing.Op(A,B)
U<-rbing.matrix.gibbs(A,B,U)

U<-rmf.matrix(Z)
U<-rmf.matrix.gibbs(Z,U)
```

`lineSearch` *A curvilinear search on the Stiefel manifold (Wen and Yin 2013, Algo 1)*

Description

A curvilinear search on the Stiefel manifold (Wen and Yin 2013, Algo 1)

Usage

```
lineSearch(F, dF, X, rho1, rho2, tauStart, maxIters = 20)
```

Arguments

`F`	A function V(n, p) -> `R^1`
`dF`	A function V(n, p) -> `R^1`
`X`	an n x p semi-orthogonal matrix (starting point)
`rho1`	Parameter for Armijo condition. Between 0 and 1 and usually small, e.g < 0.1
`rho2`	Parameter for Wolfe condition Between 0 and 1 usually large, > 0.9
`tauStart`	Initial step size
`maxIters`	Maximum number of iterations

Value

A list containing: Y, the semi-orthogonal matrix satisfying the Armijo-Wolfe conditions and tau: the stepsize satisfying these conditions

Author(s)

Alexander Franks

References

(Wen and Yin, 2013)

Examples

```
N <- 10
P <- 2
M <- diag(10:1)
F <- function(V) { - sum(diag(t(V) %*% M %*% V)) }
dF <- function(V) { - 2*M %*% V }
X <- rustiefel(N, P)
res <- lineSearch(F, dF, X, rho1=0.1, rho2=0.9, tauStart=1)
```

---

lineSearchBB — *A curvilinear search on the Stiefel manifold with BB steps (Wen and Yin 2013, Algo 2) This is based on the line search algorithm described in (Zhang and Hager, 2004)*

Description

A curvilinear search on the Stiefel manifold with BB steps (Wen and Yin 2013, Algo 2) This is based on the line search algorithm described in (Zhang and Hager, 2004)

Usage

```
lineSearchBB(F, X, Xprev, G_x, G_xprev, rho, C, maxIters = 20)
```

Arguments

F	A function V(n, p) -> R
X	an n x p semi-orthogonal matrix (the current)
Xprev	an n x p semi-orthogonal matrix (the previous)
G_x	an n x p matrix with (G_x)_ij = dF(X)/dX_ij
G_xprev	an n x p matrix with (G_xprev)_ij = dF(X_prev)/dX_prev_ij
rho	Convergence parameter, usually small (e.g. 0.1)
C	C_t+1 = (etaQ_t + F(X_t+1))/Q_t+1 See section 3.2 in Wen and Yin, 2013
maxIters	Maximum number of iterations

Value

A list containing Y: a semi-orthogonal matrix Ytau which satisfies convergence criteria (Eqn 29 in Wen & Yin '13), and tau: the stepsize satisfying these criteria

Author(s)

Alexander Franks

References

(Wen and Yin, 2013) and (Zhang and Hager, 2004)

Examples

```
N <- 10
P <- 2
M <- diag(10:1)
F <- function(V) { - sum(diag(t(V) %*% M %*% V)) }
dF <- function(V) { - 2*M %*% V }
Xprev <- rustiefel(N, P)
G_xprev <- dF(Xprev)
X <- rustiefel(N, P)
G_x <- dF(X)
Xprev <- dF(X)
res <- lineSearchBB(F, X, Xprev, G_x, G_xprev, rho=0.1, C=F(X))
```

NullC — *Null Space of a Matrix*

Description

Given a matrix `M`, find a matrix `N` giving a basis for the null space. This is a modified version of Null from the package MASS.

Usage

```
NullC(M)
```

Arguments

`M`	input matrix.

Value

an orthonormal matrix such that `t(N)%*%M` is a matrix of zeros.

Note

The MASS function `Null(matrix(0,4,2))` returns a 4*2 matrix, whereas `NullC(matrix(0,4,2))` returns `diag(4)`.

Author(s)

Peter Hoff

Examples

```
NullC(matrix(0,4,2))

## The function is currently defined as
function (M)
{
    tmp <- qr(M)
    set <- if (tmp$rank == 0L)
        1L:nrow(M)
    else -(1L:tmp$rank)
    qr.Q(tmp, complete = TRUE)[, set, drop = FALSE]
  }
```

---

optStiefel *Optimize a function on the Stiefel manifold*

Description

Find a local minimum of a function defined on the stiefel manifold using algorithms described in Wen and Yin (2013).

Usage

```
optStiefel(F, dF, Vinit, method = "bb", searchParams = NULL, tol = 1e-08,
  maxIters = 100, verbose = FALSE, maxLineSearchIters = 20)
```

Arguments

F	A function V(P, S) -> `R^1`
dF	A function to compute the gradient of F. Returns a `P x S` matrix with `dF(X)_ij = d(F(X))/dX_ij`
Vinit	The starting point on the stiefel manifold for the optimization
method	Line search type: "bb" or curvilinear
searchParams	List of parameters for the line search algorithm. If the line search algorithm is the standard curvilinear search than the search parameters are rho1 and rho2. If the line search algorithm is "bb" then the parameters are rho and eta.
tol	Convergence tolerance. Optimization stops when Fprime < abs(tol), an approximate stationary point.
maxIters	Maximum iterations for each gradient step
verbose	Boolean indicating whether to print function value and iteration number at each step.
maxLineSearchIters	Maximum iterations for for each line search (one step in the gradient descent algorithm)

Value

A stationary point of F on the Stiefel manifold.

Author(s)

Alexander Franks

References

(Wen and Yin, 2013)

Examples

```
## Find the first eigenspace spanned by the first P eigenvectors for a
## matrix M. The size of the matrix has been kept small and the tolerance
## has been raised to keep the runtime
## of this example below the CRAN submission threshold.

N <- 500
P <- 3
Lam <- diag(c(10, 5, 3, rep(1, N-P)))
U <- rustiefel(N, N)
M <- U %*% Lam %*% t(U)

F <- function(V) { - sum(diag(t(V) %*% M %*% V)) }
dF <- function(V) { - 2*M %*% V }
V = optStiefel(F, dF, Vinit=rustiefel(N, P),
               method="curvilinear",
               searchParams=list(rho1=0.1, rho2=0.9, tau=1),tol=1e-4)

print(sprintf("Sum of first %d eigenvalues is %f", P, -F(V)))
```

---

rbing.matrix.gibbs *Gibbs Sampling for the Matrix-variate Bingham Distribution*

Description

Simulate a random orthonormal matrix from the Bingham distribution using Gibbs sampling.

Usage

```
rbing.matrix.gibbs(A, B, X)
```

Arguments

A	a symmetric matrix.
B	a diagonal matrix with decreasing entries.
X	the current value of the random orthonormal matrix.

Value

a new value of the matrix X obtained by Gibbs sampling.

Note

This provides one Gibbs scan. The function should be used iteratively.

Author(s)

Peter Hoff

References

Hoff(2009)

Examples

```
Z<-matrix(rnorm(10*5),10,5) ; A<-t(Z)%*%Z
B<-diag(sort(rexp(5),decreasing=TRUE))
U<-rbing.Op(A,B)
U<-rbing.matrix.gibbs(A,B,U)

## The function is currently defined as
function (A, B, X)
{
    m <- dim(X)[1]
    R <- dim(X)[2]
    if (m > R) {
        for (r in sample(seq(1, R, length = R))) {
            N <- NullC(X[, -r])
            An <- B[r, r] * t(N) %*% (A) %*% N
            X[, r] <- N %*% rbing.vector.gibbs(An, t(N) %*% X[,
                r])
        }
    }
    if (m == R) {
        for (s in seq(1, R, length = R)) {
            r <- sort(sample(seq(1, R, length = R), 2))
            N <- NullC(X[, -r])
            An <- t(N) %*% A %*% N
            X[, r] <- N %*% rbing.Op(An, B[r, r])
        }
    }
    X
  }
```

rbing.O2 *Simulate a 2*2 Orthogonal Random Matrix*

Description

Simulate a 2*2 random orthogonal matrix from the Bingham distribution using a rejection sampler.

Usage

```
rbing.O2(A, B, a = NULL, E = NULL)
```

Arguments

A	a symmetric matrix.
B	a diagonal matrix with decreasing entries.
a	sum of the eigenvalues of A, multiplied by the difference in B-values.
E	eigenvectors of A.

Value

A random 2x2 orthogonal matrix simulated from the Bingham distribution.

Author(s)

Peter Hoff

References

Hoff(2009)

Examples

```
## The function is currently defined as
function (A, B, a = NULL, E = NULL)
{
    if (is.null(a)) {
        trA <- A[1, 1] + A[2, 2]
        lA <- 2 * sqrt(trA^2/4 - A[1, 1] * A[2, 2] + A[1, 2]^2)
        a <- lA * (B[1, 1] - B[2, 2])
        E <- diag(2)
        if (A[1, 2] != 0) {
            E <- cbind(c(0.5 * (trA + lA) - A[2, 2], A[1, 2]),
                c(0.5 * (trA - lA) - A[2, 2], A[1, 2]))
            E[, 1] <- E[, 1]/sqrt(sum(E[, 1]^2))
            E[, 2] <- E[, 2]/sqrt(sum(E[, 2]^2))
        }
    }
```

```
    b <- min(1/a^2, 0.5)
    beta <- 0.5 - b
    lrmx <- a
    if (beta > 0) {
        lrmx <- lrmx + beta * (log(beta/a) - 1)
    }
    lr <- -Inf
    while (lr < log(runif(1))) {
        w <- rbeta(1, 0.5, b)
        lr <- a * w + beta * log(1 - w) - lrmx
    }
    u <- c(sqrt(w), sqrt(1 - w)) * (-1)^rbinom(2, 1, 0.5)
    x1 <- E %*% u
    x2 <- (x1[2:1] * c(-1, 1) * (-1)^rbinom(1, 1, 0.5))
    cbind(x1, x2)
  }
```

`rbing.Op` *Simulate a* `p*p` *Orthogonal Random Matrix*

Description

Simulate a `p*p` random orthogonal matrix from the Bingham distribution using a rejection sampler.

Usage

```
rbing.Op(A, B)
```

Arguments

`A`	a symmetric matrix.
`B`	a diagonal matrix with decreasing entries.

Value

A random pxp orthogonal matrix simulated from the Bingham distribution.

Note

This only works for small matrices, otherwise the sampler will reject too frequently to be useful.

Author(s)

Peter Hoff

References

Hoff(2009)

Examples

```
Z<-matrix(rnorm(10*5),10,5) ; A<-t(Z)%*%Z
B<-diag(sort(rexp(5),decreasing=TRUE))
U<-rbing.Op(A,B)
U<-rbing.matrix.gibbs(A,B,U)


## The function is currently defined as
function (A, B)
{
    b <- diag(B)
    bmx <- max(b)
    bmn <- min(b)
    if(bmx>bmn)
    {
    A <- A * (bmx - bmn)
    b <- (b - bmn)/(bmx - bmn)
    vlA <- eigen(A)$val
    diag(A) <- diag(A) - vlA[1]
    vlA <- eigen(A)$val
    nu <- max(dim(A)[1] + 1, round(-vlA[length(vlA)]))
    del <- nu/2
    M <- solve(diag(del, nrow = dim(A)[1]) - A)/2
    rej <- TRUE
    cholM <- chol(M)
    nrej <- 0
    while (rej) {
        Z <- matrix(rnorm(nu * dim(M)[1]), nrow = nu, ncol = dim(M)[1])
        Y <- Z %*% cholM
        tmp <- eigen(t(Y) %*% Y)
        U <- tmp$vec %*% diag((-1)^rbinom(dim(A)[1], 1, 0.5))
        L <- diag(tmp$val)
        D <- diag(b) - L
        lrr <- sum(diag((D %*% t(U) %*% A %*% U))) - sum(-sort(diag(-D)) *
            vlA)
        rej <- (log(runif(1)) > lrr)
        nrej <- nrej + 1
    }
    }
    if(bmx==bmn) { U<-rustiefel(dim(A)[1],dim(A)[1]) }
    U
  }
```

`rbing.vector.gibbs` *Gibbs Sampling for the Vector-variate Bingham Distribution*

Description

Simulate a random normal vector from the Bingham distribution using Gibbs sampling.

Usage

```
rbing.vector.gibbs(A, x)
```

Arguments

A	a symmetric matrix.
x	the current value of the random normal vector.

Value

a new value of the vector x obtained by Gibbs sampling.

Note

This provides one Gibbs scan. The function should be used iteratively.

Author(s)

Peter Hoff

References

Hoff(2009)

Examples

```
## The function is currently defined as
rbing.vector.gibbs <-
function(A,x)
{
  #simulate from the vector bmf distribution as described in Hoff(2009)
  #this is one Gibbs step, and must be used iteratively
  evdA<-eigen(A,symmetric=TRUE)
  E<-evdA$vec
  l<-evdA$val

  y<-t(E)%*%x
  x<-E%*%ry_bing(y,l)
  x/sqrt(sum(x^2))
  #One improvement might be a rejection sampler
  #based on a mixture of vector mf distributions.
  #The difficulty is finding the max of the ratio.
}
```

`rbmf.matrix.gibbs` *Gibbs Sampling for the Matrix-variate Bingham-von Mises-Fisher Distribution.*

Description

Simulate a random orthonormal matrix from the Bingham distribution using Gibbs sampling.

Usage

```
rbmf.matrix.gibbs(A, B, C, X)
```

Arguments

A	a symmetric matrix.
B	a diagonal matrix with decreasing entries.
C	a matrix with the same dimension as X.
X	the current value of the random orthonormal matrix.

Value

a new value of the matrix `X` obtained by Gibbs sampling.

Note

This provides one Gibbs scan. The function should be used iteratively.

Author(s)

Peter Hoff

References

Hoff(2009)

Examples

```
## The function is currently defined as
function (A, B, C, X)
{
    m <- dim(X)[1]
    R <- dim(X)[2]
    if (m > R) {
        for (r in sample(seq(1, R, length = R))) {
            N <- NullC(X[, -r])
            An <- B[r, r] * t(N) %*% (A) %*% N
            cn <- t(N) %*% C[, r]
```

```
            X[, r] <- N %*% rbmf.vector.gibbs(An, cn, t(N) %*%
                X[, r])
        }
    }
    if (m == R) {
        for (s in seq(1, R, length = R)) {
            r <- sort(sample(seq(1, R, length = R), 2))
            N <- NullC(X[, -r])
            An <- t(N) %*% A %*% N
            Cn <- t(N) %*% C[, r]
            X[, r] <- N %*% rbmf.O2(An, B[r, r], Cn)
        }
    }
    X
  }
```

`rbmf.O2` *Simulate a* `2*2` *Orthogonal Random Matrix*

Description

Simulate a `2*2` random orthogonal matrix from the Bingham-von Mises-Fisher distribution using a rejection sampler.

Usage

```
rbmf.O2(A, B, C, env = FALSE)
```

Arguments

`A`	a symmetric matrix.
`B`	a diagonal matrix with decreasing entries.
`C`	a 2x2 matrix.
`env`	which rejection envelope to use, Bingham (`bingham`) or von Mises-Fisher (`mf`)?

Value

A random 2x2 orthogonal matrix simulated from the Bingham-von Mises-Fisher distribution.

Author(s)

Peter Hoff

References

Hoff(2009)

Examples

```
## The function is currently defined as
function (A, B, C, env = FALSE)
{
    sC <- svd(C)
    d1 <- sum(sC$d)
    eA <- eigen(A)
    ab <- sum(eA$val * diag(B))
    if (d1 <= ab | env == "bingham") {
        lrmx <- sum(sC$d)
        lr <- -Inf
        while (lr < log(runif(1))) {
            X <- rbing.O2(A, B, a = (eA$val[1] - eA$val[2]) *
                (B[1, 1] - B[2, 2]), E = eA$vec)
            lr <- sum(diag(t(X) %*% C)) - lrmx
        }
    }
    if (d1 > ab | env == "mf") {
        lrmx <- sum(eA$val * sort(diag(B), decreasing = TRUE))
        lr <- -Inf
        while (lr < log(runif(1))) {
            X <- rmf.matrix(C)
            lr <- sum(diag(B %*% t(X) %*% A %*% X)) - lrmx
        }
    }
    X
  }
```

---

rbmf.vector.gibbs *Gibbs Sampling for the Vector-variate Bingham-von Mises-Fisher Distribution*

Description

Simulate a random normal vector from the Bingham-von Mises-Fisher distribution using Gibbs sampling.

Usage

```
rbmf.vector.gibbs(A, c, x)
```

Arguments

A	a symmetric matrix.
c	a vector with the same length as x.
x	the current value of the random normal vector.

Value

a new value of the vector x obtained by Gibbs sampling.

Note

This provides one Gibbs scan. The function should be used iteratively.

Author(s)

Peter Hoff

References

Hoff(2009)

Examples

```
## The function is currently defined as
function (A, c, x)
{
    evdA <- eigen(A)
    E <- evdA$vec
    l <- evdA$val
    y <- t(E) %*% x
    d <- t(E) %*% c
    x <- E %*% ry_bmf(y, l, d)
    x/sqrt(sum(x^2))
  }
```

---

rmf.matrix *Simulate a Random Orthonormal Matrix*

Description

Simulate a random orthonormal matrix from the von Mises-Fisher distribution.

Usage

```
rmf.matrix(M)
```

Arguments

M	a matrix.

Value

an orthonormal matrix of the same dimension as M.

Author(s)

Peter Hoff

References

Hoff(2009)

Examples

```
## The function is currently defined as
Z<-matrix(rnorm(10*5),10,5)

U<-rmf.matrix(Z)
U<-rmf.matrix.gibbs(Z,U)


function (M)
{
    if (dim(M)[2] == 1) {
        X <- rmf.vector(M)
    }
    if (dim(M)[2] > 1) {
        svdM <- svd(M)
        H <- svdM$u %*% diag(svdM$d)
        m <- dim(H)[1]
        R <- dim(H)[2]
        cmet <- FALSE
        rej <- 0
        while (!cmet) {
            U <- matrix(0, m, R)
            U[, 1] <- rmf.vector(H[, 1])
            lr <- 0
            for (j in seq(2, R, length = R - 1)) {
                N <- NullC(U[, seq(1, j - 1, length = j - 1)])
                x <- rmf.vector(t(N) %*% H[, j])
                U[, j] <- N %*% x
                if (svdM$d[j] > 0) {
                  xn <- sqrt(sum((t(N) %*% H[, j])^2))
                  xd <- sqrt(sum(H[, j]^2))
                  lbr <- log(besselI(xn, 0.5 * (m - j - 1), expon.scaled = TRUE)) -
                    log(besselI(xd, 0.5 * (m - j - 1), expon.scaled = TRUE))
                  if (is.na(lbr)) {
                    lbr <- 0.5 * (log(xd) - log(xn))
                  }
                  lr <- lr + lbr + (xn - xd) + 0.5 * (m - j -
                    1) * (log(xd) - log(xn))
                }
            }
            cmet <- (log(runif(1)) < lr)
            rej <- rej + (1 - 1 * cmet)
        }
```

```
      X <- U %*% t(svd(M)$v)
    }
    X
  }
```

`rmf.matrix.gibbs` *Gibbs Sampling for the Matrix-variate von Mises-Fisher Distribution*

Description

Simulate a random orthonormal matrix from the matrix von Mises-Fisher distribution using Gibbs sampling.

Usage

```
rmf.matrix.gibbs(M, X, rscol = NULL)
```

Arguments

`M`	a matrix.
`X`	the current value of the random orthonormal matrix.
`rscol`	the number of columns to update simultaneously.

Value

a new value of the matrix `X` obtained by Gibbs sampling.

Note

This provides one Gibbs scan. The function should be used iteratively.

Author(s)

Peter Hoff

References

Hoff(2009)

Examples

```
Z<-matrix(rnorm(10*5),10,5)

U<-rmf.matrix(Z)
U<-rmf.matrix.gibbs(Z,U)


## The function is currently defined as
function (M, X, rscol = NULL)
{
    if (is.null(rscol)) {
        rscol <- max(2, min(round(log(dim(M)[1])), dim(M)[2]))
    }
    sM <- svd(M)
    H <- sM$u %*% diag(sM$d)
    Y <- X %*% sM$v
    m <- dim(H)[1]
    R <- dim(H)[2]
    for (iter in 1:round(R/rscol)) {
        r <- sample(seq(1, R, length = R), rscol)
        N <- NullC(Y[, -r])
        y <- rmf.matrix(t(N) %*% H[, r])
        Y[, r] <- N %*% y
    }
    Y %*% t(sM$v)
  }
```

`rmf.vector` *Simulate a Random Normal Vector*

Description

Simulate a random normal vector from the von Mises-Fisher distribution as described in Wood(1994).

Usage

```
rmf.vector(kmu)
```

Arguments

`kmu`	a vector.

Value

a vector.

Author(s)

Peter Hoff

References

Wood(1994), Hoff(2009)

Examples

```
## The function is currently defined as
function (kmu)
{
    kap <- sqrt(sum(kmu^2))
    mu <- kmu/kap
    m <- length(mu)
    if (kap == 0) {
        u <- rnorm(length(kmu))
        u<-matrix(u/sqrt(sum(u^2)),m,1)
    }
    if (kap > 0) {
        if (m == 1) {
            u <- (-1)^rbinom(1, 1, 1/(1 + exp(2 * kap * mu)))
        }
        if (m > 1) {
            W <- rW(kap, m)
            V <- rnorm(m - 1)
            V <- V/sqrt(sum(V^2))
            x <- c((1 - W^2)^0.5 * t(V), W)
            u <- cbind(NullC(mu), mu) %*% x
        }
    }
    u
  }
```

`rustiefel` *Siumlate a Uniformly Distributed Random Orthonormal Matrix*

Description

Siumlate a random orthonormal matrix from the uniform distribution on the Stiefel manifold.

Usage

```
rustiefel(m, R)
```

Arguments

m	the length of each column vector.
R	the number of column vectors.

Value

an m*R orthonormal matrix.

Author(s)

Peter Hoff

References

Hoff(2007)

Examples

```
## The function is currently defined as
function (m, R)
{
    X <- matrix(rnorm(m * R), m, R)
    tmp <- eigen(t(X) %*% X)
    X %*% (tmp$vec %*% sqrt(diag(1/tmp$val, nrow = R)) %*% t(tmp$vec))
  }
```

---

rW *Simulate W as Described in Wood(1994)*

Description

Auxilliary variable simulation for rejection sampling of rmf.vector, as described in Wood(1994).

Usage

```
rW(kap, m)
```

Arguments

kap	a positive scalar.
m	a positive integer.

Value

a number between zero and one.

Author(s)

Peter Hoff

Examples

```
rW(pi,4)

## The function is currently defined as
function (kap, m)
{
    .C("rW", kap = as.double(kap), m = as.integer(m), w = double(1))$w
  }
```

---

ry_bing *Helper Function for Sampling a Bingham-distributed Vector*

Description

C interface to perform a Gibbs update of `y` with invariant distribution proportional to `exp( sum(l*y^2)` with respect to the uniform measure on the sphere.

Usage

```
ry_bing(y, l)
```

Arguments

y	a normal vector.
l	a vector.

Value

a normal vector.

Author(s)

Peter Hoff

References

Hoff(2009)

Examples

```
## The function is currently defined as
function (y, l)
{
    .C("ry_bing", y = as.double(y), l = as.double(l), n = as.integer(length(y)))$y
  }
```

---

ry_bmf *Helper Function for Sampling a Bingham-von Mises-Fisher-distributed Vector*

Description

C interface to perform a Gibbs update of `y` with invariant distribution proportional to `exp( sum(l*y^2+y*d)` with respect to the uniform measure on the sphere.

Usage

```
ry_bmf(y, l, d)
```

Arguments

`y`	a normal vector.
`l`	a vector.
`d`	a vector.

Value

a normal vector

Author(s)

Peter Hoff

References

Hoff(2009)

Examples

```
## The function is currently defined as
function (y, l, d)
{
    .C("ry_bmf", y = as.double(y), l = as.double(l), d = as.double(d),
        n = as.integer(length(y)))$y
  }
```

`tr` *Compute the trace of a matrix*

Description

compute the trace of a square matrix

Usage

```
tr(X)
```

Arguments

`X`	Square matrix

Index

∗ **package**
rstiefel-package, 2

lineSearch, 3
lineSearchBB, 4

NullC, 5

optStiefel, 6

rbing.matrix.gibbs, 7
rbing.O2, 9
rbing.Op, 10
rbing.vector.gibbs, 11
rbmf.matrix.gibbs, 13
rbmf.O2, 14
rbmf.vector.gibbs, 15
rmf.matrix, 16
rmf.matrix.gibbs, 18
rmf.vector, 19
rstiefel *(*rstiefel-package*)*, 2
rstiefel-package, 2
rustiefel, 20
rW, 21
ry_bing, 22
ry_bmf, 23

tr, 24