Package: mniw (via r-universe)

October 31, 2024

Type Package

Title The Matrix-Normal Inverse-Wishart Distribution

Version 1.0.2

Date 2024-09-19

Description Density evaluation and random number generation for the Matrix-Normal Inverse-Wishart (MNIW) distribution, as well as the the Matrix-Normal, Matrix-T, Wishart, and Inverse-Wishart distributions. Core calculations are implemented in a portable (header-only) C++ library, with matrix manipulations using the 'Eigen' library for linear algebra. Also provided is a Gibbs sampler for Bayesian inference on a random-effects model with multivariate normal observations.

URL `https://github.com/mlysy/mniw/`, `https://mlysy.github.io/mniw/`

BugReports `https://github.com/mlysy/mniw/issues`

License GPL-3

Depends R (>= 2.10)

Imports Rcpp (>= 0.11.6)

LinkingTo Rcpp, RcppEigen

LazyData true

Suggests testthat, knitr, rmarkdown

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

Repository https://mlysy.r-universe.dev

RemoteUrl https://github.com/mlysy/mniw

RemoteRef HEAD

RemoteSha ef05408960e06308037ea9fcdf1c5e54aefba6e3

Contents

`mniw-package` *Tools for the Matrix-Normal Inverse-Wishart distribution.*

Description

Density evaluation and random number generation for the Matrix-Normal Inverse-Wishart (MNIW) distribution, as well as its constituent distributions, i.e., the Matrix-Normal, Matrix-T, Wishart, and Inverse-Wishart distributions.

Details

The Matrix-Normal Inverse-Wishart (MNIW) distribution $(\boldsymbol{X}, \boldsymbol{V}) \sim \text{MNIW}(\boldsymbol{\Lambda}, \boldsymbol{\Sigma}, \boldsymbol{\Psi}, \nu)$ on random matrices $\boldsymbol{X}_{p\times q}$ and symmetric positive-definite $\boldsymbol{V}_{q\times q}$ is defined as

$$\begin{array}{rcl} \boldsymbol{V} & \sim & \text{Inverse-Wishart}(\boldsymbol{\Psi}, \nu) \\ \boldsymbol{X} \mid \boldsymbol{V} & \sim & \text{Matrix-Normal}(\boldsymbol{\Lambda}, \boldsymbol{\Sigma}, \boldsymbol{V}), \end{array}$$

where the Matrix-Normal distribution is defined as the multivariate normal

$$\text{vec}(\boldsymbol{X}) \sim \mathcal{N}(\text{vec}(\boldsymbol{\Lambda}), \boldsymbol{V} \otimes \boldsymbol{\Sigma}),$$

where $\text{vec}(\boldsymbol{X})$ is a vector stacking the columns of $\boldsymbol{X}$, and $\boldsymbol{V} \otimes \boldsymbol{\Sigma}$ denotes the Kronecker product.

Author(s)

Maintainer: Martin Lysy `<mlysy@uwaterloo.ca>`

Authors:

- Bryan Yates

See Also

Useful links:

- `https://github.com/mlysy/mniw/`
- `https://mlysy.github.io/mniw/`
- Report bugs at `https://github.com/mlysy/mniw/issues`

`crossprodV` *Matrix cross-product.*

Description

Vectorized matrix cross-products `t(X) V Y` or `t(X) V^{-1} Y`.

Usage

```
crossprodV(X, Y = NULL, V, inverse = FALSE)
```

Arguments

`X`	A matrix of size `p x q`, or an array of size `p x q x n`.
`Y`	A matrix of size `p x r`, or an array of size `p x r x n`. If missing defaults to `Y = X`.
`V`	A matrix of size `p x p`, or an array of size `p x p x n`.
`inverse`	Logical; whether or not the inner product should be calculated with `V` or `V^{-1}`.

Value

An array of size `q x r x n`.

Examples

```
# problem dimensions
p <- 4
q <- 2
r <- 3
n <- 5
X <- array(rnorm(p*q*n), dim = c(p, q, n)) # vectorized
Y <- array(rnorm(p*r*n), dim = c(p, r, n)) # vectorized
V <- crossprod(matrix(rnorm(p*p), p, p)) # not vectorized (but positive definite)
crossprodV(X = X, V = V) # self cross-product
# cross-product with inverse matrix weight
crossprodV(X = X, V = V, Y = Y, inverse = TRUE)
```

`Hospitals` *Hospital profiling data.*

Description

Information on patient-reported problem rates for 27 teaching hospitals and private academic health centers in the United States.

Usage

```
Hospitals
```

Format

A data frame with 27 rows (one for each hospital) and 4 variables:

`NSrg` Non-surgery related problem rate (percent).

`Srg` Surgery related problem rate (percent).

`Severity` Average health index for surveyed patients.

`Size` Number of patients surveyed.

References

Everson, P.J. and Morris, C.N. "Inference for multivariate normal hierarchical models." *Journal of the Royal Statistical Society, Series B* 62:2 (2000): 399-412.

`MatrixNormal-dist` *The Matrix-Normal distribution.*

Description

Density and random sampling for the Matrix-Normal distribution.

Usage

```
dMNorm(X, Lambda, SigmaR, SigmaC, log = FALSE)

rMNorm(n, Lambda, SigmaR, SigmaC)
```

Arguments

`X`	Argument to the density function. Either a `p x q` matrix or a `p x q x n` array.
`Lambda`	Mean parameter. Either a `p x q` matrix or a `p x q x n` array.
`SigmaR`	Between-row covariance matrix. Either a `p x p` matrix or a `p x p x n` array.
`SigmaC`	Between-column covariance matrix Either a `q x q` matrix or a `q x q x n` array.
`log`	Logical; whether or not to compute the log-density.
`n`	Integer number of random samples to generate.

Details

The Matrix-Normal distribution $\boldsymbol{X} \sim \text{Matrix-Normal}(\boldsymbol{\Lambda}, \boldsymbol{\Sigma}_R, \boldsymbol{\Sigma}_C)$ on the random matrix $\boldsymbol{X}_{p\times q}$ is defined as

$$\text{vec}(\boldsymbol{X}) \sim \mathcal{N}(\text{vec}(\boldsymbol{\Lambda}), \boldsymbol{\Sigma}_C \otimes \boldsymbol{\Sigma}_R),$$

where $\text{vec}(\boldsymbol{X})$ is a vector stacking the columns of $\boldsymbol{X}$, and $\boldsymbol{\Sigma}_C \otimes \boldsymbol{\Sigma}_R$ denotes the Kronecker product.

Value

A vector length `n` for density evaluation, or an array of size `p x q x n` for random sampling.

Examples

```
# problem dimensions
p <- 4
q <- 2
n <- 10 # number of observations
# parameter values
Lambda <- matrix(rnorm(p*q),p,q) # mean matrix
# row-wise variance matrix (positive definite)
SigmaR <- crossprod(matrix(rnorm(p*p), p, p))
SigmaC <- rwish(n, Psi = diag(q), nu = q + 1) # column-wise variance (vectorized)

# random sample
X <- rMNorm(n, Lambda = Lambda, SigmaR = SigmaR, SigmaC = SigmaC)

# log-density at each sampled value
dMNorm(X, Lambda = Lambda, SigmaR = SigmaR, SigmaC = SigmaC, log = TRUE)
```

`MatrixT-dist` *The Matrix-t distribution.*

Description

Density and sampling for the Matrix-t distribution.

Usage

```
dMT(X, Lambda, SigmaR, SigmaC, nu, log = FALSE)

rMT(n, Lambda, SigmaR, SigmaC, nu)
```

Arguments

`X`	Argument to the density function. Either a `p x q` matrix or a `p x q x n` array.
`Lambda`	Mean parameter. Either a `p x q` matrix or a `p x q x n` array.
`SigmaR`	Between-row covariance matrix. Either a `p x p` matrix or a `p x p x n` array.
`SigmaC`	Between-column covariance matrix Either a `q x q` matrix or a `q x q x n` array.
`nu`	Degrees-of-freedom parameter. A scalar or vector of length `n`.
`log`	Logical; whether or not to compute the log-density.
`n`	Integer number of random samples to generate.

Details

The Matrix-T distribution $\boldsymbol{X} \sim \text{Matrix-T}(\boldsymbol{\Lambda}, \boldsymbol{\Sigma}, \boldsymbol{\Psi}, \nu)$ on a random matrix $\boldsymbol{X}_{p \times q}$ is the marginal distribution of $\boldsymbol{X}$ in $(\boldsymbol{X}, \boldsymbol{V}) \sim \text{MNIW}(\boldsymbol{\Lambda}, \boldsymbol{\Sigma}, \boldsymbol{\Psi}, \nu)$, where the Matrix-Normal Inverse-Wishart (MNIW) distribution is defined in MNIW-dist.

Value

A vector length `n` for density evaluation, or an array of size `p x q x n` for random sampling.

`MNIW-dist` *Generate samples from the Matrix-Normal Inverse-Wishart distribution.*

Description

Generate samples from the Matrix-Normal Inverse-Wishart distribution.

Usage

```
rMNIW(n, Lambda, Sigma, Psi, nu, prec = FALSE)

rmniw(n, Lambda, Omega, Psi, nu)
```

Arguments

`n`	number of samples.
`Lambda`	A mean matrix of size `p x q` or an array of size `p x q x n`. Defaults to matrix of zeros when missing.
`Sigma`	A row-wise variance or precision matrix of size `p x p`, or an array of size `p x p x n`. Defaults to the identity matrix when missing.
`Psi`	A scale matrix of size `q x q`, or an array of size `q x q x n`. Defaults to identity matrix when missing.
`nu`	Scalar degrees-of-freedom parameter.
`prec`	Logical; whether or not `Sigma` is on the variance or precision scale.
`Omega`	A between-row precision matrix of size `p x p`, or an array of size `p x p x n`. Defaults to the identity matrix when missing.

Details

The Matrix-Normal Inverse-Wishart (MNIW) distribution $(\boldsymbol{X}, \boldsymbol{V}) \sim \text{MNIW}(\boldsymbol{\Lambda}, \boldsymbol{\Sigma}, \boldsymbol{\Psi}, \nu)$ on random matrices $\boldsymbol{X}_{p \times q}$ and symmetric positive-definite $\boldsymbol{V}_{q \times q}$ is defined as

$$
\begin{aligned}
\boldsymbol{V} &\sim \text{Inverse-Wishart}(\boldsymbol{\Psi}, \nu) \\
\boldsymbol{X} \mid \boldsymbol{V} &\sim \text{Matrix-Normal}(\boldsymbol{\Lambda}, \boldsymbol{\Sigma}, \boldsymbol{V}),
\end{aligned}
$$

where the Matrix-Normal distribution is defined as the multivariate normal

$$\mathrm{vec}(\boldsymbol{X}) \sim \mathcal{N}(\mathrm{vec}(\boldsymbol{\Lambda}), \boldsymbol{V} \otimes \boldsymbol{\Sigma}),$$

where $\mathrm{vec}(\boldsymbol{X})$ is a vector stacking the columns of $\boldsymbol{X}$, and $\boldsymbol{V} \otimes \boldsymbol{\Sigma}$ denotes the Kronecker product.

`rmniw()` is a convenience wrapper to `rMNIW(Sigma = Omega, prec = TRUE)`, for the common situation in Bayesian inference with conjugate priors when between-row variances are naturally parametrized on the precision scale.

Value

A list with elements:

`X`	Array of size `p x q x n` random samples from the Matrix-Normal component (see Details).
`V`	Array of size `q x q x n` of random samples from the Inverse-Wishart component.

Examples

```
# problem dimensions
p <- 2
q <- 3
n <- 10 # number of samples
# parameter specification
Lambda <- matrix(rnorm(p*q),p,q) # single argument
Sigma <- rwish(n, Psi = diag(p), nu = p + rexp(1)) # vectorized argument
Psi <- rwish(n = 1, Psi = diag(q), nu = q + rexp(1)) # single argument
nu <- q + rexp(1)
# simulate n draws
rMNIW(n, Lambda = Lambda, Sigma = Sigma, Psi = Psi, nu = nu)
```

`MultiNormal-dist` *The Multivariate Normal distribution.*

Description

Density and random sampling for the Multivariate Normal distribution.

Usage

```
dmNorm(x, mu, Sigma, log = FALSE)

rmNorm(n, mu, Sigma)
```

Arguments

`x`	Argument to the density function. A vector of length `q` or an `n x q` matrix.
`mu`	Mean vector(s). Either a vector of length `q` or an `n x q` matrix. If missing defaults to a vector of zeros.
`Sigma`	Covariance matrix or matrices. Either a `q x q` matrix or a `q x q x n` array. If missing defaults to the identity matrix.
`log`	Logical; whether or not to compute the log-density.
`n`	Integer number of random samples to generate.

Value

A vector for densities, or a `n x q` matrix for random sampling.

Examples

```
# Parameter specification
q <- 4 # number of dimensions
mu <- 1:q # mean vector
V <- toeplitz(exp(-seq(1:q))) # variance matrix

# Random sample
n <- 100
X <- rmNorm(n, mu, V)

# Calculate log density for each sampled vector
dmNorm(X, mu, V, log = TRUE)
```

`rRxNorm` *Conditional sampling for Multivariate-Normal Random-Effects model.*

Description

Sample from the conditional parameter distribution given the data and hyperparameters of the Multivariate-Normal Random-Effects (mNormRE) model (see **Details**).

Usage

```
rRxNorm(n, x, V, lambda, Sigma)
```

Arguments

`n`	Integer number of random samples to generate.
`x`	Data observations. Either a vector of length `q` or a `n x q` matrix. In the latter case each row is a different vector.
`V`	Observation variances. Either a matrix of size `q x q` or a `q x q x n` array.

`lambda`	Prior means. Either a vector of length `q` or an `n x q` matrix. In the latter case each row is a different mean. Defaults to zeros.
`Sigma`	Prior variances. Either a matrix of size `q x q` or a `q x q x n` array. Defaults to identity matrix.

Details

Consider the hierarchical multivariate normal model

$$\begin{aligned} \boldsymbol{\mu} &\sim \mathcal{N}(\boldsymbol{\lambda}, \boldsymbol{\Sigma}) \\ \boldsymbol{x} \mid \boldsymbol{\mu} &\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{V}). \end{aligned}$$

The Multivariate-Normal Random-Effects model $\boldsymbol{\mu} \sim \mathrm{RxNorm}(\boldsymbol{x}, \boldsymbol{V}, \boldsymbol{\lambda}, \boldsymbol{\Sigma})$ on the random vector $\boldsymbol{\mu}_q$ is defined as the posterior distribution $p(\boldsymbol{\mu} \mid \boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\Sigma})$. This distribution is multivariate normal; for the mathematical specification of its parameters please see `vignette("mniw-distributions", package = "mniw")`.

Examples

```
# data specification
q <- 5
y <- rnorm(q)
V <- rwish(1, diag(q), q+1)
# prior specification
lambda <- rep(0,q)
A <- diag(q)
n <- 10
# random sampling
rRxNorm(n, y, V, lambda, A)
```

`RxNormLM` *Bayesian inference for a random-effects regression model.*

Description

Gibbs sampler for posterior distribution of parameters and hyperparameters of a multivariate normal random-effects linear regression model called RxNormLM (see Details).

Usage

```
RxNormLM(
  nsamples,
  Y,
  V,
  X,
  prior = NULL,
  init,
  burn,
  updateHyp = TRUE,
```

```
  storeHyp = TRUE,
  updateRX = TRUE,
  storeRX = FALSE
)
```

Arguments

`nsamples`	number of posterior samples to draw.
`Y`	`N x q` matrix of responses.
`V`	Either a `q x q` variance matrix or an `q x q x N` array of such matrices.
`X`	`N x p` matrix of covariates.
`prior`	parameters of the prior MNIW distribution on the hyperparameters (see Details).
`init`	(optional) list with elements `Beta`, `Sigma`, and `Mu` providing the initial values for these. Default values are `Beta = matrix(0, p, q)`, `Sigma = diag(q)`, and `Mu = Y`.
`burn`	integer number of burn-in samples, or fraction of `nsamples` to prepend as burn-in.
`updateHyp`, `storeHyp`	logical. Whether or not to update/store the hyperparameter draws.
`updateRX`, `storeRX`	logical. Whether or not to update/store the random-effects draws.

Details

The RxNormLM model is given by

$$y_i \mid \mu_i \sim_i idN(\mu_i, V_i)$$

$$\mu_i \mid \beta, \Sigma\ sim_i ndN(x_i'\beta, \Sigma)$$

$$\beta, \Sigma\ MNIW(\Lambda, \Omega^{-1}, \Psi, \nu),$$

where y_i and μ_i are response and random-effects vectors of length q, x_i are covariate vectors of length p, and (β, Σ) are hyperparameter matrices of size $p \times q$ and $q \times q$.

The MNIW prior distribution is given by a list with elements `Lambda`, `Omega`, `Psi`, and `nu`. If any of these is `NULL` or missing, the default value is 0. Note that `Omega == 0` gives a Lebesgue prior to β.

Value

A list with (potential) elements:

`Beta` An `p x q x nsamples` array of regression coefficient iterations (if `storeHyp == TRUE`)

`Sigma` An `q x q x nsamples` array of regression variance matrices (if `storeHyp == TRUE`)

`Mu` An `n x q x nsamples` array of random effects (if `storeRX == TRUE`)

Examples

```
# problem dimensions
n <- sample(10:20,1) # number of observations
p <- sample(1:4,1) # number of covariates
q <- sample(1:4,1) # number of responses

# hyperparameters
Lambda <- rMNorm(1, Lambda = matrix(0, p, q))
Omega <- crossprod(rMNorm(1, Lambda = matrix(0, p, p)))
Psi <- crossprod(rMNorm(1, Lambda = matrix(0, q, q)))
nu <- rexp(1) + (q+1)
prior <- list(Lambda = Lambda, Omega = Omega, Psi = Psi, nu = nu)

# random-effects parameters
BSig <- rmniw(1, Lambda = Lambda, Omega = Omega, Psi = Psi, nu = nu)
Beta <- BSig$X
Sigma <- BSig$V

# design matrix
X <- rMNorm(1, matrix(0, n, p))

# random-effects themselves
Mu <- rmNorm(n, X %*% Beta, Sigma)

# generate response data
V <- rwish(n, Psi = diag(q), nu = q+1) # error variances
Y <- rmNorm(n, mu = Mu, Sigma = V) # responses

# visual checks for each component of Gibbs sampler

# sample from p(Mu | Beta, Sigma, Y)
nsamples <- 1e5
out <- RxNormLM(nsamples,
                Y = Y, V = V, X = X,
                prior = prior,
                init = list(Beta = Beta, Sigma = Sigma, Mu = Mu),
                burn = floor(nsamples/10),
                updateHyp = FALSE,
                storeHyp = FALSE,
                updateRX = TRUE,
                storeRX = TRUE)

# conditional distribution is RxNorm:
iObs <- sample(n, 1) # pick an observation at random
# calculate the RxNorm parameters
G <- Sigma %*% solve(V[,,iObs] + Sigma)
xB <- c(X[iObs,,drop=FALSE] %*% Beta)
muRx <- G %*% (Y[iObs,] - xB) + xB
SigmaRx <- G %*% V[,,iObs]

# a' * mu_i is univariate normal with known mean and variance:
a <- rnorm(q) # arbitrary vector
```

```
amui <- crossprod(a, out$Mu[iObs,,]) # a' * mu_i

hist(amui, breaks = 100, freq = FALSE,
     xlab = "", main = expression("Histogram of "*a^T*mu[i]))
curve(dnorm(x, mean = sum(a * muRx),
            sd = sqrt(crossprod(a, SigmaRx %*% a)[1])),
      add = TRUE, col = "red")
legend("topright",
       legend = c("Observed", "Expected"),
       lwd = c(NA, 2), pch = c(22, NA), seg.len = 1.5,
       col = c("black", "red"), bg = c("white", NA))



# sample from p(Beta, Sigma | Mu, Y)
nsamples <- 1e5
out <- RxNormLM(nsamples,
                Y = Y, V = V, X = X,
                prior = prior,
                init = list(Beta = Beta, Sigma = Sigma, Mu = Mu),
                burn = floor(nsamples/10),
                updateHyp = TRUE,
                storeHyp = TRUE,
                updateRX = FALSE,
                storeRX = FALSE)


# conditional distribution is MNIW:
# calculate the MNIW parameters
OmegaHat <- crossprod(X) + Omega
LambdaHat <- solve(OmegaHat, crossprod(X, Mu) + Omega %*% Lambda)
PsiHat <- Psi + crossprod(Mu) + crossprod(Lambda, Omega %*% Lambda)
PsiHat <- PsiHat - crossprod(LambdaHat, OmegaHat %*% LambdaHat)
nuHat <- nu + n

# a' Sigma^{-1} a is chi^2 with known parameters:
a <- rnorm(q)
aSiga <- drop(crossprodV(a, V = out$Sigma, inverse = TRUE))
sigX <- crossprod(a, solve(PsiHat, a))[1]
hist(aSiga, breaks = 100, freq = FALSE,
     xlab = "", main = expression("Histogram of "*a^T*Sigma^{-1}*a))
curve(dchisq(x/sigX, df = nuHat)/sigX, add = TRUE, col = "red")
legend("topright",
       legend = c("Observed", "Expected"),
       lwd = c(NA, 2), pch = c(22, NA), seg.len = 1.5,
       col = c("black", "red"), bg = c("white", NA))

# a' Beta b is student-t with known parameters:
a <- rnorm(p)
b <- rnorm(q)
# vectorized calculations
aBetab <- crossprodV(X = aperm(out$Beta, c(2,1,3)),
                     Y = b, V = diag(q)) # Beta b
```

```
aBetab <- drop(crossprodV(X = a, Y = aBetab, V = diag(p))) # a' Beta b
# student-t parameters
muT <- crossprod(a, LambdaHat %*% b)[1]
nuT <- nuHat-q+1
sigmaT <- crossprodV(a, V = OmegaHat, inverse = TRUE)[1]
sigmaT <- sigmaT * crossprodV(b, V = PsiHat)[1]
sigmaT <- sqrt(sigmaT / nuT)

hist(aBetab, breaks = 100, freq = FALSE,
     xlab = "", main = expression("Histogram of "*a^T*Beta*a))
curve(dt((x-muT)/sigmaT, df = nuT)/sigmaT, add = TRUE, col = "red")
legend("topright",
       legend = c("Observed", "Expected"),
       lwd = c(NA, 2), pch = c(22, NA), seg.len = 1.5,
       col = c("black", "red"), bg = c("white", NA))
```

`Wishart-dist` *Wishart and Inverse-Wishart distributions.*

Description

Densities and random sampling for the Wishart and Inverse-Wishart distributions.

Usage

```
dwish(X, Psi, nu, log = FALSE)

rwish(n, Psi, nu)

diwish(X, Psi, nu, log = FALSE)

riwish(n, Psi, nu)

dwishart(X, Psi, nu, inverse = FALSE, log = FALSE)

rwishart(n, Psi, nu, inverse = FALSE)
```

Arguments

`X`	Argument to the density function. Either a `q x q` matrix or a `q x q x n` array.
`Psi`	Scale parameter. Either a `q x q` matrix or a `q x q x n` array.
`nu`	Degrees-of-freedom parameter. A scalar or vector of length `n`.
`log`	Logical; whether or not to compute the log-density.
`n`	Integer number of random samples to generate.
`inverse`	Logical; whether or not to use the Inverse-Wishart distribution.

Details

The Wishart distribution $\boldsymbol{X} \sim \text{Wishart}(\boldsymbol{\Psi}, \nu)$ on a symmetric positive-definite random matrix $\boldsymbol{X}$ of size $q \times q$ has PDF

$$f(\boldsymbol{X} \mid \boldsymbol{\Psi}, \nu) = \frac{|\boldsymbol{X}|^{(\nu-q-1)/2} \exp\big\{ - \text{tr}(\boldsymbol{\Psi}^{-1}\boldsymbol{X})/2\big\}}{2^{q\nu/2}|\boldsymbol{\Psi}|^{\nu/2}\Gamma_q(\frac{\nu}{2})},$$

where $\Gamma_q(\alpha)$ is the multivariate gamma function,

$$\Gamma_q(\alpha) = \pi^{q(q-1)/4} \prod_{i=1}^{q} \Gamma(\alpha + (1-i)/2).$$

The Inverse-Wishart distribution $\boldsymbol{X} \sim \text{Inverse-Wishart}(\boldsymbol{\Psi}, \nu)$ is defined as $\boldsymbol{X}^{-1} \sim \text{Wishart}(\boldsymbol{\Psi}^{-1}, \nu)$.

`dwish()` and `diwish()` are convenience wrappers for `dwishart()`, and similarly `rwish()` and `riwish()` are wrappers for `rwishart()`.

Value

A vector length `n` for density evaluation, or an array of size `q x q x n` for random sampling.

Examples

```
# Random sampling

n <- 1e5
q <- 3
Psi1 <- crossprod(matrix(rnorm(q^2),q,q))
nu <- q + runif(1, 0, 5)

X1 <- rwish(n,Psi1,nu) # Wishart

# plot it
plot_fun <- function(X) {
  q <- dim(X)[1]
  par(mfrow = c(q,q))
  for(ii in 1:q) {
    for(jj in 1:q) {
      hist(X[ii,jj,], breaks = 100, freq = FALSE,
           xlab = "", main = parse(text = paste0("X[", ii, jj, "]")))
    }
  }
}

plot_fun(X1)

# "vectorized" scale parameeter
Psi2 <- 5 * Psi1
vPsi <- array(c(Psi1, Psi2), dim = c(q, q, n))
X2 <- rwish(n, Psi = vPsi, nu = nu)
plot_fun(X2)
```

```
# Inverse-Wishart
X3 <- riwish(n, Psi2, nu)
plot_fun(X3)

# log-density calculation for sampled values
par(mfrow = c(1,1))
hist(dwish(X2, vPsi, nu, log = TRUE),
     breaks = 100, freq = FALSE, xlab = "",
     main = expression("log-p"*(X[2]*" | "*list(Psi,nu))))
```

Index