Package 'myFunctions'

February 25, 2015

Type Package

Title Implements my commonly used functions

Version 1.0

Date 2015-02-24

Author John Tipton

Maintainer John Tipton <jtipton25@gmail.comt>

Description Implements my commonly used functions

License GPL (>= 2)

Imports Rcpp (>= 0.11.4)

LinkingTo Rcpp, RcppArmadillo

R topics documented:

`myFunctions-package` *A set of commonly used R and C++ functions*

Description

A set of commonly used R and C++ functions

Details

Package:	myFunctions
Type:	Package
Version:	1.0
Date:	2015-02-24
License:	What license is it under?

~~ An overview of how to use the package, including the most important functions ~~

Author(s)

John Tipton

Maintainer: John Tipton <jtipton25@gmail.com> ~~ The author and/or maintainer of the package ~~

References

~~ Literature or other references for background information ~~

See Also

~~ Optional links to other man pages, e.g. ~~ ~~ <pkg> ~~

Examples

```
~~ Examples ~~
```

`RcppArmadillo-Functions` *Binds two matrices together into a matrix*

Description

Binds two matrices with the same number of rows together into a common matrix in a fashion similar to cbind in R

Usage

```
cbindARMA(A, B)
```

Arguments

A	a numeric matrix
B	a numeric matrix

Details

This function takes two matrices and binds them together into a single matrix.

Value

`cbindARMA()` returns a matrix with the same number of rows as both `A` and `B` and the number of columns of `A` plus the number of columns `B` giving a matrix, with some abuse of notation of the form `[A B]`. Note that there is no way to deal with missing values in this function and this requires the number of rows of `A` to be the same as the number of rows of `B` (If this isn't true, behavior of the function is not guaranteed).

Author(s)

John Tipton

References

None

Examples

```
A <- matrix(1:4, 2, 2)
B <- matrix(5:8, 2, 2)

cbindARMA(A, B)
```

```
##       [,1] [,2] [,3] [,4]
##  [1,]    1    3    5    7
##  [2,]    2    4    6    8
```

RcppArmadillo-Functions

Calculates column sums

Description

Calculates column sums

Usage

```
colSums(x)
```

Arguments

x	a numeric matrix

Details

These functions take a matrix and return a vector of column sums.

Value

`colSums()` returns a vector which gives the column sums of the matrix `x`. Note that there is no way to deal with missing values in this function.

Author(s)

John Tipton

References

None

Examples

```
  x <- matrix(1:4, 2, 2)
  colSums(x)
##        [,1]
##  [1,]    3
##  [2,]    7
```

`RcppArmadillo-Functions` *Moment matching of mean and variance to shape and rate parameters of inverse gamma function*

Description

Matches the mean and varianceto shape and rate parameters of inverse gamma function

Usage

```
convertToAlpha(mu, s2)
convertToBeta(mu, s2)
```

Arguments

`mu > 0`	a double
`s2 > 0`	a double

Details

These functions take a mean and variance parameterization of an inverse gamma distribution and return `rate` and `shape` parameters of the inverse gamma distribution.

Value

convertToAlpha() returns a double that represents the shape parameter of an inverse gamma distribution with mean mu and variance s2. convertToBeta() returns a double that represents the rate parameter of an inverse gamma distribution with mean mu and variance s2. Note this returns a shape > 2 and a rate > 0s.

Author(s)

John Tipton

References

None

Examples

```
  set.seed(101)
  mu <- 4  ## must be greater than 0
  s2 <- 2  ## must be greater than 0
  rate <- convertToAlpha(mu, s2)
  shape <- convertToBeta(mu, s2)

  x <- 1 / rgamma(10000, rate, shape)
  mean(x)
##  [1] 3.994432
  var(x)
##  [1] 1.978426
```

RcppArmadillo-Functions

Moment matching of mean and variance to shape and rate parameters of inverse gamma function

---

Description

Matches the mean and varianceto shape and rate parameters of inverse gamma function

Usage

```
convertToAlpha(mu, s2)
convertToBeta(mu, s2)
```

Arguments

mu > 0	a double
s2 > 0	a double

Details

These functions take a mean and variance parameterization of an inverse gamma distribution and return rate and shape parameters of the inverse gamma distribution.

Value

convertToAlpha() returns a double that represents the shape parameter of an inverse gamma distribution with mean mu and variance s2. convertToBeta() returns a double that represents the rate parameter of an inverse gamma distribution with mean mu and variance s2. Note this returns a shape > 2 and a rate > 0s.

Author(s)

John Tipton

References

None

Examples

```
  set.seed(101)
  mu <- 4  ## must be greater than 0
  s2 <- 2  ## must be greater than 0
  rate <- convertToAlpha(mu, s2)
  shape <- convertToBeta(mu, s2)

  x <- 1 / rgamma(10000, rate, shape)
  mean(x)
##  [1] 3.994432
  var(x)
##  [1] 1.978426
```

RcppArmadillo-Functions

Multivariate normal density evaluation <- Not sure why I have dmvnormArmaVec

---

Description

Matches the mean and varianceto shape and rate parameters of inverse gamma function

Usage

```
dMVNorm(x, mu, Sig)
dmvnormArmaVec(x, mu, Sig, logd = FALSE)
```

Arguments

x	a numeric vector
mu	a numeric vector
Sig	a numeric matrix
logd	a boolean, if TRUE returns the log density

Details

These functions take a vector realization `x` from a random multivariate normal variable with mean vector `mu` and covariance matrix `Sig` and returns the log density of that random variable.

Value

`dMVNorm()` returns a double that represents the log denisty of a multivariate random normal variable. `dmvnormArmaVec()` returns a numeric vector that the log denisty of a multivariate random normal variable.

Author(s)

John Tipton

References

Wikipedia...

Examples

```
  mu <- 3:4
  Sig <- matrix(c(1, 0.5, 0.5, 1), 2, 2)
  x <- 4:5

  dMVNorm(x, mu, Sig)
##  [1] -2.360703

  dmvnormArmaVec(x, t(mu), Sig)
##            [,1]
##  [1,] -2.360703
##  [2,] -2.360703
```

`RcppArmadillo-Functions`

C++ version of svd

Description

Creates a singular value decomposition

Usage

```
svdARMA(X)
dcsvdARMA(X)
```

Arguments

`X`	a numeric matrix

Details

Multivariate normal sampler for Bayesian full conditionals.

Value

`svdARMA()` generates a singular value decomposition of `X`. `dcsvdARMA()` generates a singular value decomposition of `X` using the divide and conquer algorithm.

`U`	a numeric matrix of left singular vectors
`sd`	a numeric vector containing the singular values (square roots of the eigenvalues)
`V`	a numeric matrix of right singular vectors

Author(s)

John Tipton

References

None

Examples

```
  X <- matrix(1:4, 2, 2)
  Sig <- t(X) %*% X

  svdARMA(Sig)
##  $sd
##             [,1]
##  [1,] 29.8660687
##  [2,]  0.1339313
##
##  $U
##             [,1]       [,2]
##  [1,] -0.4045536 -0.9145143
##  [2,] -0.9145143  0.4045536
##
##  $V
##             [,1]       [,2]
##  [1,] -0.4045536 -0.9145143
##  [2,] -0.9145143  0.4045536

  dcsvdARMA(Sig)
##  $sd
##             [,1]
##  [1,] 29.8660687
##  [2,]  0.1339313
##
##  $U
##             [,1]       [,2]
##  [1,] -0.4045536 -0.9145143
##  [2,] -0.9145143  0.4045536
```

RcppArmadillo-Functions

Density of inverse gamma distribution

Description

Density of inverse gamma distribution

Usage

```
dinvgammaArma(x, shape, rate, logarithm = true)
dinvgammaArmaVec(y, shape, rate, logarithm = true)
```

Arguments

x	a double
y	a numeric vector
shape	a double
rate	a double
logarithm	a boolean

Details

These functions take a double or vector and return a log inverse gamma density evaluation using the parameterization

$$\frac{rate^{shape}}{\gamma(shape) * x^{shape+1}} exp(-rate/x)$$

.

Value

`dinvgammaArma()` returns a double which is the log density of an inverse gamma distribtiuon.

`dinvgammaArmaVec()` returns a numeric vector where each element of the vector is the log density of an inverse gamma distribtuion.

Author(s)

John Tipton

References

See wikipedia...

Examples

```
x <- 0.5
y <- 1:4
rate <- 1
shape <- 1
dinvgammaArma(x, 1, 1) ## returns log density
dinvgammaArmaVec(y, 1, 1) ## returns vector of log densities
```

RcppArmadillo-Functions

Density of inverse gamma distribution

Description

Density of inverse gamma distribution

Usage

```
dinvgammaArma(x, shape, rate, logarithm = true)
dinvgammaArmaVec(y, shape, rate, logarithm = true)
```

Arguments

x	a double
y	a numeric vector
shape	a double
rate	a double
logarithm	a boolean

Details

These functions take a double or vector and return a log inverse gamma density evaluation using the parameterization

Value

`dinvgammaArma()` returns a double which is the log density of an inverse gamma distribtiuon.

`dinvgammaArmaVec()` returns a numeric vector where each element of the vector is the log density of an inverse gamma distribtuion.

Author(s)

John Tipton

References

See wikipedia...

Examples

```
x <- 0.5
y <- 1:4
rate <- 1
shape <- 1
dinvgammaArma(x, 1, 1) ## returns log density
dinvgammaArmaVec(y, 1, 1) ## returns vector of log densities
```

RcppArmadillo-Functions

Multivariate normal density evaluation <- Not sure why I have dmvnormArmaVec??

Description

Matches the mean and varianceto shape and rate parameters of inverse gamma function

Usage

```
dMVNorm(x, mu, Sig)
dmvnormArmaVec(x, mu, Sig, logd = FALSE)
```

Arguments

x	a numeric vector
mu	a numeric vector
Sig	a numeric matrix
logd	a boolean, if TRUE returns the log density

Details

These functions take a vector realization x from a random multivariate normal variable with mean vector mu and covariance matrix Sig and returns the log density of that random variable.

Value

dMVNorm() returns a double that represents the log denisty of a multivariate random normal variable. dmvnormArmaVec() returns a numeric vector that the log denisty of a multivariate random normal variable.

Author(s)

John Tipton

References

Wikipedia...

Examples

```
  mu <- 3:4
  Sig <- matrix(c(1, 0.5, 0.5, 1), 2, 2)
  x <- 4:5

  dMVNorm(x, mu, Sig)
##  [1] -2.360703

  dmvnormArmaVec(x, t(mu), Sig)
##            [,1]
##  [1,] -2.360703
```

```
##  [2,] -2.360703
```

RcppArmadillo-Functions

Fast approximate normal CDF

Description

Fast approximation of the CDF of a normal random variable

Usage

```
phi(x)
```

Arguments

x	a double

Details

This function is a fast approximation of pnorm in C++ that loses accuracy in the tails.

Value

`fastPhi()` returns a double between 0 and 1 that gives the CDF of a standard normal random variable.

Author(s)

John Tipton

References

None

Examples

```
  x <- 1.64

  pnorm(x)
##  [1] 0.9494974

  phi(x)
##  [1] 0.9494974
```

RcppArmadillo-Functions *log determinant of covariance matrix*

Description

Calculate the log determinant of a positive definite (covariance) matrix

Usage

```
logDet(Sig)
```

Arguments

Sig	a positive definite matrix

Details

Gives the log determinant of positive definite matrix.

Value

`logDet()` returns a double that gives the log determinant of a positive definite matrix.

Author(s)

John Tipton

References

None

Examples

```
  X <- matrix(1:4, 2, 2)
  Sig <- t(X) %*% X

  logDet(Sig)
## [1] 1.386294
```

RcppArmadillo-Functions *Calculates Euclidean distance between two sets of coordinates*

Description

Calculates Euclidean distance between two sets of coordinates

Usage

```
makeDistARMA(coords1, coords)
```

Arguments

coords1	a numeric matrix with two columns
coords2	a numeric matrix with two columns

Details

Gives the distance matrix between coords1 and coords2.

Value

makeDistARMA() returns a matrix that gives the distances between coords1 and coords2.

Author(s)

John Tipton

References

None

Examples

```
  coords1 <- matrix(1:8, 4, 2)
  coords2 <- matrix(11:18, 4, 2)

  makeDistARMA(coords1, coords2)
##          [,1]     [,2]     [,3]     [,4]
##  [1,] 0.000000 1.414214 2.828427 4.242641
##  [2,] 1.414214 0.000000 1.414214 2.828427
##  [3,] 2.828427 1.414214 0.000000 1.414214
##  [4,] 4.242641 2.828427 1.414214 0.000000
```

RcppArmadillo-Functions

C++ version of princomp

Description

Calculates principal component matrix

Usage

```
makePCA(X)
```

Arguments

X	a numeric matrix

Details

Gives the rotated principal components and associated standard deviations of the eigenvalues of the matrix X as in princomp in R.

Value

makePCA() returns a matrix that gives the principal components and the standard deviations of the eigenvalues, as in princomp in R with the following components:

X_pca	a numeric matrix whose columns contain the eigenvalues
sdev	a vector of the standard deviations of the principal components

Author(s)

John Tipton

References

None

Examples

```
  X <- matrix(1:8, 4, 2)

  makePCA(X)
##  $X_pca
##             [,1] [,2]
##  [1,] -2.1213203    0
##  [2,] -0.7071068    0
##  [3,]  0.7071068    0
##  [4,]  2.1213203    0
##
##  $sdev
##           [,1]
##  [1,] 1.825742
##  [2,] 0.000000
```

RcppArmadillo-Functions

C++ version of mvrnorm

Description

Simulates a multivariate normal random variable

Usage

```
mvrnormArma(n, mu, Sigma)
mvrnormArmaVec(mu, Sigma)
```

Arguments

n	an integer, the number of samples
mu	a numeric mean vector
Sigma	a positive definite covariance matrix

Details

Gives draws from a multivariate normal random variable.

Value

mvrnormArma() returns a matrix that gives a sample from a multivariate normal random variable with mean mu and covariance Sigma in each row. mvrnormArmaVec() returns a vector that is a single sample from a multivariate normal random variable with mean mu and covariance Sigma in each column.

Author(s)

John Tipton

References

None

Examples

```
  mu <- 1:4
  Sigma <- diag(4)

  mvrnormArma(4, mu, Sigma)
##              [,1]     [,2]     [,3]     [,4]
##  [1,]  1.32247945 2.537651 1.472034 4.268270
##  [2,]  0.07755503 1.530883 2.845135 3.698179
##  [3,] -0.56829123 1.395040 4.632903 5.262550
##  [4,]  0.17389172 2.670689 3.337295 1.686998

  mvrnormArmaVec(mu, Sigma)
##            [,1]
```

```
##  [1,] -1.571272
##  [2,]  3.124006
##  [3,]  5.760263
##  [4,]  6.286767
```

RcppArmadillo-Functions

C++ version of mvrnorm

Description

Simulates a multivariate normal random variable

Usage

```
mvrnormArma(n, mu, Sigma)
mvrnormArmaVec(mu, Sigma)
```

Arguments

n	an integer, the number of samples
mu	a numeric mean vector
Sigma	a positive definite covariance matrix

Details

Gives draws from a multivariate normal random variable.

Value

mvrnormArma() returns a matrix that gives a sample from a multivariate normal random variable with mean mu and covariance Sigma in each row. mvrnormArmaVec() returns a vector that is a single sample from a multivariate normal random variable with mean mu and covariance Sigma in each column.

Author(s)

John Tipton

References

None

Examples

```
mu <- 1:4
Sigma <- diag(4)

mvrnormArma(4, mu, Sigma)
##              [,1]     [,2]     [,3]     [,4]
##  [1,]  1.32247945 2.537651 1.472034 4.268270
##  [2,]  0.07755503 1.530883 2.845135 3.698179
```

```
##  [3,] -0.56829123 1.395040 4.632903 5.262550
##  [4,]  0.17389172 2.670689 3.337295 1.686998

  mvrnormArmaVec(mu, Sigma)
##            [,1]
##  [1,] -1.571272
##  [2,]  3.124006
##  [3,]  5.760263
##  [4,]  6.286767
```

RcppArmadillo-Functions

C++ version of order

Description

Creates an order index as in order in R

Usage

```
orderArma(x)
```

Arguments

x	a numeric vector

Details

Values are sorted in descending order.

Value

`orderArma()` returns a permutation index as in order in R.

Author(s)

John Tipton

References

None

Examples

```
  x <- c(4,7,3,5,9,2)

  orderArma(x)
##  [1] 6 3 1 4 2 5

  x[orderArma(x)]
##  [1] 2 3 4 5 7 9
```

RcppArmadillo-Functions

C++ version of order

Description

Creates an order index as in order in R

Usage

```
rMVNArma(A, x)
rMVNArmaScalar(a, b)
```

Arguments

A	a positive definite precision matrix A
x	a numeric vector
a	a double
b	a double

Details

Multivariate normal sampler for Bayesian full conditionals.

Value

`rMVNArma()` generates samples from a multivariate normal distribution with mean `A^{-1} %*% x` and covariance matrix `A^{-1}`. `rMVNArmaScalar()` generates samples from a multivariate normal distribution with mean

Author(s)

John Tipton

References

None

Examples

```
set.seed(101)
x <- 1:4
A <- diag(4)
a <- 2
b <- 1

rMVNArma(A, x)
##            [,1]
##  [1,] 0.8746786
##  [2,] 1.5526778
##  [3,] 4.2836179
##  [4,] 4.9653297
```

```
  rMVNArmaScalar(a, b)
##  [1] 0.02274263
```

RcppArmadillo-Functions

C++ version of order

Description

Creates an order index as in order in R

Usage

```
rMVNArma(A, x)
rMVNArmaScalar(a, b)
```

Arguments

A	a positive definite precision matrix A
x	a numeric vector
a	a double
b	a double

Details

Multivariate normal sampler for Bayesian full conditionals.

Value

`rMVNArma()` generates samples from a multivariate normal distribution with mean `A^{-1} %*% x` and covariance matrix `A^{-1}`. `rMVNArmaScalar()` generates samples from a multivariate normal distribution with mean

Author(s)

John Tipton

References

None

Examples

```
  set.seed(101)
  x <- 1:4
  A <- diag(4)
  a <- 2
  b <- 1

  rMVNArma(A, x)
##            [,1]
```

```
##  [1,] 0.8746786
##  [2,] 1.5526778
##  [3,] 4.2836179
##  [4,] 4.9653297

  rMVNArmaScalar(a, b)
##  [1] 0.02274263
```

RcppArmadillo-Functions

Stacks two matrices together into a matrix

Description

Stacks two matrices with the same number of columns together into a common matrix in a fashion similar to rbind in R

Usage

```
rbindARMA(X, Y)
```

Arguments

X	a numeric matrix
Y	a numeric matrix

Details

This function takes two matrices and return a stacked matrix.

Value

`rbindARMA()` returns a matrix with the same number of columns as both `X` and `Y` and the number of rows of `X` plus the number of rows `Y` giving a matrix, with some abuse of notation of the form Note that there is no way to deal with missing values in this function and this requires the number of columns of `X` to be the same as the number of columns of `Y` (If this isn't true, behavior of the function is not guaranteed).

Author(s)

John Tipton

References

None

Examples

```
  X <- matrix(1:4, 2, 2)
  Y <- matrix(5:8, 2, 2)

  rbindARMA(X, Y)
##      [,1] [,2]
##  [1,]    1    3
##  [2,]    2    4
##  [3,]    5    7
##  [4,]    6    8
```

RcppArmadillo-Functions

Calculates row means

Description

Calculates row means

Usage

```
rowMeans(x)
```

Arguments

x	a numeric matrix

Details

These functions take a matrix and return a vector of row means.

Value

`rowMeans()` returns a vector which gives the row means of the matrix `x`. Note that there is no way to deal with missing values in this function.

Author(s)

John Tipton

References

None

Examples

```
  x <- matrix(1:4, 2, 2)
  rowMeans(x)
##      [,1]
##  [1,]    2
##  [2,]    3
```

RcppArmadillo-Functions
Calculates row standard deviations

Description

Calculates row standard deviations

Usage

```
rowSds(x)
```

Arguments

x	a numeric matrix

Details

These functions take a matrix and return a vector of row standard deviations.

Value

`rowSds()` returns a vector which gives the row standard deviations of the matrix `x`. Note that there is no way to deal with missing values in this function.

Author(s)

John Tipton

References

None

Examples

```
  x <- matrix(1:4, 2, 2)
  rowSds(x)
##         [,1]
##  [1,] 1.414214
##  [2,] 1.414214
```

RcppArmadillo-Functions *Calculates row sums*

Description

Calculates row sums

Usage

```
rowSums(X)
```

Arguments

X	a numeric matrix

Details

This function takes a matrix and returns a vector of row sums.

Value

`rowSums()` returns a vector which gives the row sums of the matrix `X`. Note that there is no way to deal with missing values in this function.

Author(s)

John Tipton

References

None

Examples

```
  X <- matrix(1:4, 2, 2)

  rowSums(X)
##       [,1]
##  [1,]    4
##  [2,]    6
```

RcppArmadillo-Functions

C++ version of svd

Description

Creates a singular value decomposition

Usage

```
svdARMA(X)
dcsvdARMA(X)
```

Arguments

X	a numeric matrix

Details

Multivariate normal sampler for Bayesian full conditionals.

Value

`svdARMA()` generates a singular value decomposition of `X`. `dcsvdARMA()` generates a singular value decomposition of `X` using the divide and conquer algorithm.

U	a numeric matrix of left singular vectors
sd	a numeric vector containing the singular values (square roots of the eigenvalues)
V	a numeric matrix of right singular vectors

Author(s)

John Tipton

References

None

Examples

```
X <- matrix(1:4, 2, 2)
Sig <- t(X) %*% X

svdARMA(Sig)
##  $sd
##             [,1]
##  [1,] 29.8660687
##  [2,]  0.1339313
##
##  $U
##             [,1]       [,2]
##  [1,] -0.4045536 -0.9145143
##  [2,] -0.9145143  0.4045536
```

```
##
##  $V
##             [,1]       [,2]
##  [1,] -0.4045536 -0.9145143
##  [2,] -0.9145143  0.4045536

  dcsvdARMA(Sig)
##  $sd
##             [,1]
##  [1,] 29.8660687
##  [2,]  0.1339313
##
##  $U
##             [,1]       [,2]
##  [1,] -0.4045536 -0.9145143
##  [2,] -0.9145143  0.4045536
```

Index