

# Package ‘Homework1’

November 13, 2013

**Type** Package

**Title** PH140.778 Homework1

**Version** 1.0

**Date** 2013-11-13

**Author** Lu Li

**Maintainer** Lu Li <lli48@jhu.edu>

**Description** This package gives faster algorithms to fit a linear regression model and compute the multivariate normal density.

**License** GPL

## R topics documented:

Homework1-package . . . . .	1
dmvnorm . . . . .	2
fastlm . . . . .	3

<b>Index</b>	<b>6</b>
--------------	----------

---

Homework1-package	<i>PH140.778 Homework1</i>
-------------------	----------------------------

---

## Description

This package gives faster algorithms to fit a linear regression model and compute the multivariate normal density.

## Details

Package: Homework1  
Type: Package  
Version: 1.0  
Date: 2013-11-13  
License: GPL

This package mainly uses Cholesky decomposition when computing the inverse matrix and takes advantages of matrix properties as well.

### Author(s)

Lu Li

Maintainer Lu Li <lli48@jhu.edu>

### References

PH140.778 Advanced Statistical Computing Dr.Peng

---

dmvnorm

*Computing the Multivariate Normal Density MORE Efficiently*

---

### Description

This function dmvnorm() evaluates the k-dimensional multivariate Normal density with mean mu and covariance S

### Usage

```
dmvnorm(x, mu, S, log = TRUE)
```

### Arguments

x	a n*k matrix of points to be evaluated
mu	a length k vector of means for the k-dimensional Normal
S	a k*k covariance matrix
log	whether the return should be the log density. Default is TRUE

### Value

This function returns a vector of length n containing the values of the multivariate Normal density evaluated at the n points. If log = TRUE, it returns the log density at those points. If log is NOT TRUE, it will return the density values.

### Author(s)

Lu Li

### References

PH140.778 Advanced Statistical Computing Dr.Peng

## Examples

```
dmvnorm <- function(x, mu, S, log=TRUE) {
  #k dimension multivariate normal
  k=length(mu)
  if(is.matrix(x)==FALSE){
    x=as.matrix(t(x))
  }
  #n data points
  n=nrow(x)

  #check positive definite by trying to Cholesky decomposition
  Q=tryCatch({chol(S)},
    error=function(li){
      message("S cannot be a covariance matrix")
    })

  #compute Q_inverse*(x-mu)
  #note that t(x-mu)%*%inv(Q)%*%t(Q))%*%(x-mu)=crossprod(inv(Q)%*%(x-mu))
  #the easiest way to compute that is to solve t(Q)%*%(inv(Q)%*%(x-mu))=x-mu
  A=forwardsolve(t(Q),t(x)-mu)
  temp2=diag(crossprod(A))

  #compute density
  density=(-k/2)*log(2*pi)-sum(log(diag(Q)))-(1/2)*temp2

  #check if log argument
  if(log!=TRUE){
    density=exp(density)
  }
  return(density)
}

n <- 10
n2 <- n^2
xg <- seq(0, 1, length = n)
yg <- xg
g <- data.matrix(expand.grid(xg, yg))
D <- as.matrix(dist(g))
phi <- 5

S <- exp(-phi * D)
mu <- rep(0, n2)
set.seed(1)
x <- matrix(rnorm(n2), byrow = TRUE, ncol = n2)

dmvnorm(x, mu, S, log = TRUE)
```

## Description

Comparing to the `lm.fit()` function in R, this `fastlm()` function presents a much faster way to fit a linear regression model.

**Usage**

```
fastlm(X, y, na.rm = FALSE)
```

**Arguments**

X	predictor variable, a n*k matrix
y	response variable, a vector of length n
na.rm	argument indicating that whether missing values in X or y should be removed. Default is FALSE

**Details**

fastlm() takes the advantages of Cholesky decomposition to compute the inverse matrices, which makes a huge improvement in the efficiency of fitting linear regression models.

**Value**

a list of components

coefficients	a vector of the regression coefficients estimated using maximum likelihood
vcov	the p*p covariance matrix of the estimated regression coefficients

**Author(s)**

Lu Li

**References**

PH140.778 Advanced Statistical Computing Dr.Peng

**Examples**

```
fastlm<-function(X, y, na.rm=FALSE) {
  #check argument na.rm
  if (na.rm!=FALSE) {
    Z=cbind(X,y)
    X=X[complete.cases(Z),]
    y=as.matrix(y)[complete.cases(Z)]
  }
  n<-length(y)
  p<-ncol(X)
  #calculating transpose(X)%*%X
  A<-crossprod(X)
  #calculating transpose(X)%*%y
  C<-crossprod(X,y)

  #cholesky decomposition
  Q<-chol(A)

  #solve betahat
  temp1<-forwardsolve(t(Q),C)
  betahat<-backsolve(Q,temp1)
```

```

#calculate covirance of beta
#note that  $t(e) * e = t(e) * y - t(y) * X * \beta_{\text{hat}}$ 

cov_beta <- chol2inv(Q) * as.numeric(crossprod(y) - crossprod(X * beta_hat)) / (n - p)

return(list(coefficients = beta_hat, vcov = cov_beta))
}

set.seed(2)
## Generate predictor matrix
n <- 100
p <- 5
X <- cbind(1, matrix(rnorm(n * (p - 1)), n, p - 1))

## Coefficents
b <- rnorm(p)

## Response
y <- X * b + rnorm(n)

fit <- fastlm(X, y)
str(fit)

```

# Index

- \*Topic **Cholesky decomposition**
  - fastlm, [3](#)
- \*Topic **Linear Regression Model**
  - fastlm, [3](#)
- \*Topic **package**
  - Homework1-package, [1](#)
- dmvnorm, [2](#)
- fastlm, [3](#)
- Homework1 (Homework1-package), [1](#)
- Homework1-package, [1](#)