Package ‘Homework1’

November 8, 2013

Type Package

Title Advanced Statistics Computing HW1 Fast Algorithm

Version 1.0

Date 2013-11-08

Author Yu Du<`ydu@jhsph.edu`>

Maintainer Yu Du<`ydu@jhsph.edu`>

Description
Homework 1 in Advanced Statistics Computing Class which includes two functions.One function is to fit a fast linear model and the other function is to calculate multivariate normal densities.

License GPL

R topics documented:

`Homework1-package` *Homework 1*

Description

Advanced Statistics Computing Homework 1 which includes fastlm() and dmvnorm() two functions.

Details

Package:	Homework1
Type:	Package
Version:	1.0
Date:	2013-11-08
License:	GPL

run fastlm() to fit a linear model and dmvnorm() to evaluate mvn density.

Author(s)

Yu Du

Maintainer: Yu Du<ydu@jhsph.edu>

References

Advanced Statistics Computing Class. Dr. Peng.

`dmvnorm` *Fast Multivariate Normal Density*

Description

Evaluates the k-dimensional multivariate Normal density with mean mu and covariance matrix S.

Usage

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

Arguments

`x`	x is a n-by-k matrix with each row is a sample from a k-dimensional multivariate normal distribution
`mu`	mu is the mean vector of length k for the given multivariate normal distribution.
`S`	S is the p-by-p variance covariance matrix for the given multivariate normal distribution.
`log`	if log=TRUE which is the default value, the logged density values will be returned otherwise the original density value will be returned.

Details

This function evaluates the k-dimensional multivariate Normal density with mean mu and covariance matrix S.

Value

A vector of length n of density values will be returned. n is the number of rows of input matrix x.

Author(s)

Yu Du

References

Advanced Statistics Computing Class, Dr. Peng.

Examples

```
## Create the covariance matrix
n <- 100
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)

## The function is currently defined as
dmvnorm <- function(x, mu, S, log = TRUE) {
if (!is.matrix(x)){
        x=t(as.matrix(x))
}
k=length(mu)
n=nrow(x)

##Check if S is positive definite
R=tryCatch({chol(S)},
         error=function(e){
                 message("S is not positive definite")
         })

logdetS=2*sum(log(diag(R)))
T=x-rep(1,n)
C=forwardsolve(t(R),t(T))
term3=diag(crossprod(C))
fx=(-k/2)*log(2*pi)-(1/2)*logdetS-(1/2)*term3
if(log==TRUE){
        fx=fx
}else {
        fx=exp(fx)
}
return(fx)
}
```

`fastlm` *Fast Linear Model Fitting*

Description

fit a linear regression model to outcome data y and predictor data in a matrix X.

Usage

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

Arguments

`X`	X is a design matrix including predictor data and intercept.
`y`	y is a vector of outcome data.
`na.rm`	na.rm=TRUE to remove NA values otherwise the default value is FALSE.

Details

This function fits a linear regression model to outcome data y and predictor data in a matrix X. Matrix X also includes the intercept,i.e. the design matrix. This is faster than lm.fit().

Value

`coefficients`	The coefficients returned are the fitted regression coefficient to the input dataset.
`vcov`	The vcov returned is the variance-covariance matrix of the estimated coefficients.

Author(s)

Yu Du

References

Advanced Statistics Computing Class, Dr. Peng.

Examples

```
set.seed(2)
## Generate predictor matrix
n <- 100000
p <- 500
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)


## The function is currently defined as
fastlm=function(X,y,na.rm=FALSE){
if(na.rm==TRUE){
        r=cbind(X,y)
        X=X[complete.cases(r),]
        y=as.matrix(y[complete.cases(r)])
}

##Cholesky factorization for coefficients
A=crossprod(X)
B=crossprod(X,y)
R=chol(A)
Rbeta=forwardsolve(t(R),B)
coefficients=backsolve(R,Rbeta)

##Calculate VCOV
n=length(y)
p=ncol(X)
sigmahat2=(crossprod(y)-crossprod(coefficients,B))/(n-p)
Ainv=chol2inv(R)
vcov=as.numeric(sigmahat2)*Ainv

return(list(coefficients=coefficients,vcov=vcov))
}
```

Index