Package ‘Homework1’

November 9, 2013

Type Package

Title PH140.778 Advanced Statistical Computing HW1

Version 1.0

Date 2013-11-09

Author Lu Li

Maintainer Lu Li<lli48@jhu.edu>

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

License GPL

R topics documented:

`Homework1-package` *PH140.778 Homework1*

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:	2013-11-08
License:	GPL

This package mainly uses Cholesky decomposition when computing the inverse matrix and takes advanges of matrix property 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 funtion 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

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

        }
        n=nrow(x)

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

        #compute Q_inverse*(x-mu)
        temp1=x-rep(1,n)%*%t(mu)
        A=forwardsolve(t(Q),t(temp1))
        temp2=diag(crossprod(A))

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

        #check if log argument
        if(log==FALSE){
                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)
```

`fastlm` *Faster Way to Fit Linear Regression Models*

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

Examples

```
function (X, y, na.rm = FALSE)
{
    n <- length(y)
    p <- ncol(X)

    ##Check if missing values in X and y should be removed
    if (na.rm == TRUE) {
        Z = cbind(X, y)
        X = X[complete.cases(Z), ]
        y = as.matrix(y[complete.cases(Z)])
    }
    A <- crossprod(X)
    C <- crossprod(X, y)

    ##Cholesky decomposition
    Q <- chol(A)
    temp1 <- forwardsolve(t(Q), C)
    betahat <- backsolve(Q, temp1)
    cov_beta <- chol2inv(Q) * as.numeric(crossprod(y)-crossprod(y,X%*%betahat)/(n - p))
    return(list(coeffients = betahat, 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