Package ‘Homework1’

November 13, 2013

Type Package

Title Homework1 for advanced statistical computing

Version 1.0

Date 2013-11-13

Author Stephen Cristiano

Maintainer `<scristia@jhsph.edu>`

Description fast linear regression, fast mulvariate normal density

License GPL

R topics documented:

`dmvnorm` *multivariate normal density*

Description

Quickly evaluate multivariate normal density.

Usage

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

Arguments

`x`

`mu`

`S`

`log`

Details

Uses cholesky transformation with backwards/forwards substitution to find coefficient estimates and covariance of beta_hat.

Author(s)

Stephen Cristiano

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x, mu, S, log = TRUE)
{
    if (!is.matrix(x))
        x <- matrix(x, nrow = 1, ncol = ncol(S))
    k <- ncol(S)
    U <- try(chol(S), silent = TRUE)
    if (class(U) == "try-error")
        stop("S is not positive definite")
    d <- diag(U)
    logd <- sum(log(d))
    b <- crossprod(forwardsolve(t(U), t(x - mu)))
    if (is.matrix(b))
        b <- diag(b)
    logf <- -k/2 * log(2 * pi) - logd - 0.5 * b
    if (log)
        return(logf)
    else return(exp(logf))
  }
```

fastlm *Fast linear model*

Description

~fast~ linear regression

Usage

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

Arguments

X

y

na.rm

Details

Uses Cholesky transormation.

Note

works pretty well.

Author(s)

Stephen Cristiano

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (X, y, na.rm = FALSE)
{
    if (na.rm) {
        if (any(is.na(X))) {
            inx <- which(is.na(rowSums(X)))
            X <- X[-inx, , drop = FALSE]
            y <- y[-inx]
        }
        if (any(is.na(y))) {
            iny <- which(is.na(y))
            y <- y[-iny]
            X <- X[-iny, , drop = FALSE]
        }
    }
    n <- nrow(X)
    p <- ncol(X)
    U <- chol(t(X) %*% X)
    b <- backsolve(U, forwardsolve(t(U), t(X) %*% y))
    sigma2 <- 1/(n - p) * (t(y) %*% y - t(b) %*% t(X) %*% y)
    s2 <- diag(as.numeric(sigma2), p)
    var.b <- backsolve(U, forwardsolve(t(U), s2))
    return(list(coefficients = b, vcov = var.b))
  }
```

Index