Package 'LogRegLinux'

November 29, 2012

Type Package

Title What the package does (short line)

Version 1.0

Date 2012-11-29

Author Who wrote it

Maintainer Who to complain to <yourfault@somewhere.net>

Description More about what it does (maybe more than one line)

License What license is it under?

R topics documented:

LogReg-package 1
est.bes 2
expit 3
gbd 3
logit 4
lr.loglike 5
plot.lrsurf 5
test.data 6

Index **7**

`LogReg-package` *A basic package for Logistic Regression*

Description

More about what it does (maybe more than one line) ~~ A concise (1-5 lines) description of the package ~~

Details

Package:	LogReg
Type:	Package
Version:	1.0
Date:	2012-11-29
License:	What license is it under?

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

Author(s)

Who wrote it

Maintainer: Who to complain to <yourfault@somewhere.net> ~~ 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> ~~

`est.bes` *A function to estimate betas in a logistic regression*

Usage

```
est.bes(Y, X, intercept = TRUE, start.vals = rep(0, ncol(X) + as.numeric(interce
```

Arguments

`Y`

`X`

`intercept`

`start.vals`

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 (Y, X, intercept = TRUE, start.vals = rep(0, ncol(X) +
    as.numeric(intercept)))
{
    x <- start.vals
    opt <- optim(x, fn = lr.loglike, gr = NULL, Y = Y, X = X,
```

```
        intercept = intercept, method = "BFGS", control = list(fnscale = -1),
        hessian = T)
    beta.hat <- opt$par
    beta.cov <- solve(opt$hessian)/length(Y)
    if (intercept) {
        Xm <- cbind(rep(1, length(Y)), X)
    }
    else {
        Xm <- X
    }
    fitted.lp <- Xm %*% beta.hat
    fitted.Y <- expit(fitted.lp)
    conv <- TRUE
    if (opt$convergence != 0) {
        conv <- FALSE
        print("Optim did not converge")
    }
    return(list(ests = beta.hat, varcov = beta.cov, fitted.lp = fitted.lp,
        fitted.Y = fitted.Y, conv = conv))
}
```

`expit` *The Expit Function*

Usage

```
expit(x)
```

Arguments

```
x
```

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)
{
    1/(1 + exp(-x))
  }
```

`gbd` *A function to generate some binary regression data*

Usage

```
gbd(n = 1000, p = 1, intercept = TRUE, betas = rep(0, p + intercept), mu = rep(0
```

Arguments

n

p

intercept

betas

mu

Sigma

seed

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 (n = 1000, p = 1, intercept = TRUE, betas = rep(0, p +
    intercept), mu = rep(0, p), Sigma = diag(1, p), seed = 1)
{
    require(MASS)
    set.seed(seed)
    X <- mvrnorm(n, mu, Sigma)
    Xm <- X
    if (intercept) {
        Xm <- cbind(rep(1, n), X)
    }
    lp <- Xm %*% betas
    pv <- expit(lp)
    Y <- rbinom(n, 1, pv)
    return(list(Y = Y, X = X))
  }
```

---

logit *The logit function*

Usage

```
logit(x)
```

Arguments

x

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)
```

```
{
    return(log(x/(1 - x)))
  }
```

`lr.loglike` *Evaluate the logistic regression log-likelihood*

Usage

```
lr.loglike(betas, Y, X, intercept = TRUE)
```

Arguments

```
betas
Y
X
intercept
```

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 (betas, Y, X, intercept = TRUE)
{
    if (!intercept) {
        lp <- X %*% betas
    }
    else {
        lp <- cbind(rep(1, length(Y)), X) %*% betas
    }
    pv <- expit(lp)
    return(sum(dbinom(Y, 1, pv, log = T)))
  }
```

`plot.lrsurf` *Plot the log-likelihood surface for a single predictor*

Usage

```
plot.lrsurf(Y, X, b0 = seq(-2, 2, by = 0.01), b1 = seq(-2, 2, by = 0.01), ncl =
```

Arguments

```
Y
X
b0
b1
ncl
```

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 (Y, X, b0 = seq(-2, 2, by = 0.01), b1 = seq(-2, 2, by = 0.01),
    ncl = 20)
{
    lrs <- outer(b0, b1, "+") * 0
    for (i in 1:length(Y)) {
        lpv <- outer(b0, b1 * X[i], "+")
        lrs <- lrs + lpv * Y[i] - log(1 + exp(lpv))
    }
    image(b0, b1, lrs, xlab = expression(beta[0]), ylab = expression(beta[1]))
    contour(b0, b1, lrs, add = T, nlevels = ncl)
  }
```

`test.data` *A test binary data set with two predictors*

Usage

```
data(test.data)
```

Format

A data frame with 100 observations on the following 3 variables.

`Y` a numeric vector

`X1` a numeric vector

`X2` a numeric vector

Examples

```
data(test.data)
## maybe str(test.data) ; plot(test.data) ...
```

Index

∗Topic **\textasciitildekwd1**
    est.bes, 2
    expit, 3
    gbd, 3
    logit, 4
    lr.loglike, 5
    plot.lrsurf, 5
∗Topic **\textasciitildekwd2**
    est.bes, 2
    expit, 3
    gbd, 3
    logit, 4
    lr.loglike, 5
    plot.lrsurf, 5
∗Topic **datasets**
    test.data, 6
∗Topic **package**
    LogReg-package, 1
<pkg>, *2*

est.bes, 2
expit, 3

gbd, 3

logit, 4
LogReg *(LogReg-package)*, 1
LogReg-package, 1
lr.loglike, 5

plot.lrsurf, 5

test.data, 6