Package 'Bhat'

July 21, 2025

Title General Likelihood Exploration

Version 0.9-12

Description Provides functions for Maximum Likelihood Estimation, Markov Chain Monte Carlo, finding confidence intervals. The implementation is heavily based on the original Fortran source code translated to R.

License GPL (>= 2)

Encoding UTF-8

Imports graphics, MASS, stats

RoxygenNote 7.0.2

Suggests testthat (>= 2.0.0)

Config/testthat/edition 2

NeedsCompilation no

Author Georg Luebeck [aut] (ORCID: <https://orcid.org/0000-0002-4378-2829>), Rafael Meza [aut, cre] (ORCID: <https://orcid.org/0000-0002-1076-5037>), Alexander Gaenko [ctb] (ORCID: <https://orcid.org/0000-0002-1901-0394>)

Maintainer Rafael Meza <rmeza@umich.edu>

Repository CRAN

Date/Publication 2022-05-10 13:30:02 UTC

Contents

btrf *Generalized inverse-logit transform*

Description

maps real line onto open interval (xl, xu) using the transform y = (exp(xt) * xu + xl)/(1.+exp(xt)) where xt is a numeric vector with -Inf < xt < Inf

Usage

```
btrf(xt, xl, xu)
```

Arguments

xt	a numeric vector
xl	a numeric vector of same length as x
xu	a numeric vector of same length as x, and xu > xl

Value

returns the inverse-logit transform (numeric) of xt

Author(s)

E. Georg Luebeck (FHCRC)

See Also

ftrf

dfp *Function minimization with box-constraints*

Description

This Davidon-Fletcher-Powell optimization algorithm has been 'hand-tuned' for minimal setup configuration and for efficency. It uses an internal logit-type transformation based on the pre-specified box-constraints. Therefore, it usually does not require rescaling (see help for the R optim function). dfp automatically computes step sizes for each parameter to operate with sufficient sensitivity in the functional output. Performance is comparable to the BFGS algorithm in the R function optim. dfp interfaces with newton to ascertain convergence, compute the eigenvalues of the Hessian, and provide 95% confidence intervals when the function to be minimized is a negative log-likelihood.

Usage

```
dfp(x, f, tol = 1e-05, nfcn = 0, ...)
```

Arguments

x	a list with components 'label' (of mode character), 'est' (the parameter vector with the initial guess), 'low' (vector with lower bounds), and 'upp' (vector with upper bounds)
f	the function that is to be minimized over the parameter vector defined by the list x
tol	a tolerance used to determine when convergence should be indicated
nfcn	number of function calls
...	other parameters to be passed to 'f'

Details

The dfp function minimizes a function f over the parameters specified in the input list x. The algorithm is based on Fletcher's "Switching Method" (Comp.J. 13,317 (1970))

the code has been 'transcribed' from Fortran source code into R

Value

list with the following components:

fmin	the function value f at the minimum
label	the labels taken from list x
est	a vector of the estimates at the minimum. dfp does not overwrite x
status	0 indicates convergence, 1 indicates non-convergence
nfcn	no. of function calls

Note

This function is part of the Bhat exploration tool

Author(s)

E. Georg Luebeck (FHCRC)

References

Fletcher's Switching Method (Comp.J. 13,317, 1970)

See Also

optim, newton, ftrf, btrf, logit.hessian

Examples

```
          # generate some Poisson counts on the fly
            dose <- c(rep(0,50),rep(1,50),rep(5,50),rep(10,50))
            data <- cbind(dose,rpois(200,20*(1+dose*.5*(1-dose*0.05))))

          # neg. log-likelihood of Poisson model with 'linear-quadratic' mean:
            lkh <- function (x) {
            ds <- data[, 1]
            y  <- data[, 2]
            g <- x[1] * (1 + ds * x[2] * (1 - x[3] * ds))
            return(sum(g - y * log(g)))
            }

  # for example define
          x <- list(label=c("a","b","c"),est=c(10.,10.,.01),low=c(0,0,0),upp=c(100,20,.1))

  # call:
    results <- dfp(x,f=lkh)
```

`dqstep` *step size generator*

Description

`dqstep` determines the smallest steps ds from s so that abs(f(s+ds)-f(s)) equals a pre-specified sensitivity

Usage

```
dqstep(x, f, sens)
```

Arguments

`x`	a list with components 'label' (of mode character), 'est' (the parameter vector with the initial guess), 'low' (vector with lower bounds), and 'upp' (vector with upper bounds)
`f`	the function that is to be minimized over the parameter vector defined by the list `x`
`sens`	target sensitivity (i.e. the value of f(s+ds)-f(s))

Details

uses simple quadratic interpolation

Value

returns a vector with the desired step sizes

Note

This function is part of the Bhat exploration tool

Author(s)

E. Georg Luebeck (FHCRC)

See Also

dfp, newton, logit.hessian

Examples

```
## Rosenbrock Banana function
 fr <- function(x) {
       x1 <- x[1]
       x2 <- x[2]
       100 * (x2 - x1 * x1)^2 + (1 - x1)^2
  }
## define
 x <- list(label=c("a","b"),est=c(1,1),low=c(0,0),upp=c(100,100))
 dqstep(x,fr,sens=1)
```

---

ftrf *Generalized logit transform*

Description

maps a bounded parameter x onto the real line according to y=log((x-xl)/(xu-x))), with xl < x < xu. If this constraint is violated, an error occurs. x may be vector

Usage

```
ftrf(x, xl, xu)
```

Arguments

x	a numeric vector
xl	a numeric vector of same length as x with x > xl
xu	a numeric vector of same length as x with x < xu

Value

returns numerical vector of transforms

Author(s)

E. Georg Luebeck (FHCRC)

See Also

btrf

---

global *Random search for a global function minimum*

Description

This function generates MCMC samples from a (posterior) density function f (not necessarily normalized) in search of a global minimum of f. It uses a simple Metropolis algorithm to generate the samples. Global monitors the mcmc samples and returns the minimum value of f, as well as a sample covariance (covm) that can be used as input for the Bhat function mymcmc.

Usage

```
global(
  x,
  nlogf,
  beta = 1,
  mc = 1000,
  scl = 2,
  skip = 1,
  nfcn = 0,
  plot = FALSE
)
```

Arguments

x	a list with components 'label' (of mode character), 'est' (the parameter vector with the initial guess), 'low' (vector with lower bounds), and 'upp' (vector with upper bounds)
nlogf	negative log of the density function (not necessarily normalized)
beta	'inverse temperature' parameter
mc	length of MCMC search run
scl	not used
skip	number of cycles skipped for graphical output
nfcn	number of function calls
plot	logical variable. If TRUE the chain and the negative log density (nlogf) is plotted

Details

standard output reports a summary of the acceptance fraction, the current values of nlogf and the parameters for every (100*skip) th cycle. Plotted chains show values only for every (skip) th cycle.

Value

list with the following components:

fmin	minimum value of nlogf for the samples obtained
xmin	parameter values at fmin
covm	covariance matrix of differences between consecutive samples in chain

Note

This function is part of the Bhat package

Author(s)

E. Georg Luebeck (FHCRC)

References

too numerous to be listed here

See Also

dfp, newton, logit.hessian mymcmc

Examples

```
# generate some Poisson counts on the fly
  dose <- c(rep(0,50),rep(1,50),rep(5,50),rep(10,50))
  data <- cbind(dose,rpois(200,20*(1+dose*.5*(1-dose*0.05))))

# neg. log-likelihood of Poisson model with 'linear-quadratic' mean:
  nlogf <- function (x) {
  ds <- data[, 1]
  y  <- data[, 2]
  g <- x[1] * (1 + ds * x[2] * (1 - x[3] * ds))
  return(sum(g - y * log(g)))
  }

# initialize global search
  x <- list(label=c("a","b","c"), est=c(10, 0.25, 0.05), low=c(0,0,0), upp=c(100,10,.1))
# samples from posterior density (~exp(-nlogf))) with non-informative
# (random uniform) priors for "a", "b" and "c".
out <- global(x, nlogf, beta = 1., mc=1000, scl=2, skip=1, nfcn = 0, plot=TRUE)
# start MCMC from some other point: e.g. try x$est <- c(16,.2,.02)
```

`logit.hessian` *Hessian (curvature matrix)*

Description

Numerical evaluation of the Hessian of a real function f: $R^n \rightarrow R$ on a generalized logit scale, i.e. using transformed parameters according to x'=log((x-xl)/(xu-x))), with xl < x < xu.

Usage

```
logit.hessian(
  x = x,
  f = f,
  del = rep(0.002, length(x$est)),
  dapprox = FALSE,
  nfcn = 0
)
```

Arguments

`x`	a list with components 'label' (of mode character), 'est' (the parameter vector with the initial guess), 'low' (vector with lower bounds), and 'upp' (vector with upper bounds)
`f`	the function for which the Hessian is to be computed at point x
`del`	step size on logit scale (numeric)
`dapprox`	logical variable. If TRUE the off-diagonal elements are set to zero. If FALSE (default) the full Hessian is computed
`nfcn`	number of function calls

Details

This version uses a symmetric grid for the numerical evaluation computation of first and second derivatives.

Value

returns list with

`df`	first derivatives (logit scale)
`ddf`	Hessian (logit scale)
`nfcn`	number of function calls
`eigen`	eigen values (logit scale)

Note

This function is part of the Bhat exploration tool

Author(s)

E. Georg Luebeck (FHCRC)

See Also

dfp, newton, ftrf, btrf, dqstep

Examples

```
## Rosenbrock Banana function
 fr <- function(x) {
       x1 <- x[1]
       x2 <- x[2]
       100 * (x2 - x1 * x1)^2 + (1 - x1)^2
  }
## define
 x <- list(label=c("a","b"),est=c(1,1),low=c(-100,-100),upp=c(100,100))
 logit.hessian(x,f=fr,del=dqstep(x,f=fr,sens=0.01))
## shows the differences in curvature at the minimum of the Banana
## function along principal axis (in a logit-transformed coordinate system)
```

---

mymcmc *Adaptive Multivariate MCMC sampler*

Description

This function generates MCMC-based samples from a (posterior) density f (not necessarily normalized). It uses a Metropolis algorithm in conjunction with a multivariate normal proposal distribution which is updated adaptively by monitoring the correlations of succesive increments of at least 2 pilot chains. The method is described in De Gunst, Dewanji and Luebeck (submitted). The adaptive method is similar to the one proposed in Gelfand and Sahu (JCGS 3:261–276, 1994).

Usage

```
mymcmc(
  x,
  nlogf,
  m1,
  m2 = m1,
  m3,
  scl1 = 0.5,
  scl2 = 2,
  skip = 1,
  covm = 0,
  nfcn = 0,
  plot = FALSE,
  plot.range = 0
)
```

Arguments

x	a list with components 'label' (of mode character), 'est' (the parameter vector with the initial guess), 'low' (vector with lower bounds), and 'upp' (vector with upper bounds)
nlogf	negative log of the density function (not necessarily normalized) for which samples are to be obtained
m1	length of first pilot run (not used when covm supplied)
m2	length of second pilot run (not used when covm supplied)
m3	length of final run
scl1	scale for covariance of mv normal proposal (second pilot run)
scl2	scale for covariance of mv normal proposal (final run)
skip	number of cycles skipped for graphical output
covm	covariance matrix for multivariate normal proposal distribution. If supplied, all pilot runs will be skipped and a run of length m3 will be produced. Useful to continue a simulation from a given point with specified covm
nfcn	number of function calls
plot	logical variable. If TRUE the chain and the negative log density (nlogf) is plotted. The first m1+m2 cycles are shown in green, other cycles in red
plot.range	[Not documented. Leave as default]

Details

standard output reports a summary of the acceptance fraction, the current values of nlogf and the parameters for every (100*skip) th cycle. Plotted chains show values only for every (skip) th cycle.

Value

list with the following components:

f	values of nlogf for the samples obtained
mcmc	the chain (samples obtained)
covm	current covariance matrix for mv normal proposal distribution

Note

This function is part of the Bhat exploration tool

Author(s)

E. Georg Luebeck (FHCRC)

References

too numerous to be listed here

See Also

dfp, newton, logit.hessian

Examples

```
# generate some Poisson counts on the fly
  dose <- c(rep(0,50),rep(1,50),rep(5,50),rep(10,50))
  data <- cbind(dose,rpois(200,20*(1+dose*.5*(1-dose*0.05))))

# neg. log-likelihood of Poisson model with 'linear-quadratic' mean:
  nlogf <- function (x) {
  ds <- data[, 1]
  y  <- data[, 2]
  g <- x[1] * (1 + ds * x[2] * (1 - x[3] * ds))
  return(sum(g - y * log(g)))
  }

# start MCMC near mle
  x <- list(label=c("a","b","c"), est=c(20, 0.5, 0.05), low=c(0,0,0), upp=c(100,10,.1))
# samples from posterior density (~exp(-nlogf))) with non-informative
# (random uniform) priors for "a", "b" and "c".
out <- mymcmc(x, nlogf, m1=2000, m2=2000, m3=10000, scl1=0.5, scl2=2, skip=10, plot=TRUE)
# start MCMC from some other point: e.g. try x$est <- c(16,.2,.02)
```

---

newton *Function minimization with box-constraints*

Description

Newton-Raphson algorithm for minimizing a function `f` over the parameters specified in the input list `x`. Note, a Newton-Raphson search is very efficient in the 'quadratic region' near the optimum. In higher dimensions it tends to be rather unstable and may behave chaotically. Therefore, a (local or global) minimum should be available to begin with. Use the `optim` or `dfp` functions to search for optima.

Usage

```
newton(x, f, eps = 0.1, itmax = 10, relax = 0, nfcn = 0)
```

Arguments

`x`	a list with components 'label' (of mode character), 'est' (the parameter vector with the initial guess), 'low' (vector with lower bounds), and 'upp' (vector with upper bounds)
`f`	the function that is to be minimized over the parameter vector defined by the list `x`
`eps`	converges when all (logit-transformed) derivatives are smaller `eps`

itmax	maximum number of Newton-Raphson iterations
relax	numeric. If 0, take full Newton step, otherwise 'relax' step incrementally until a better value is found
nfcn	number of function calls

Value

list with the following components:

fmin	the function value f at the minimum
label	the labels
est	a vector of the parameter estimates at the minimum. newton does not overwrite x
low	lower 95% (Wald) confidence bound
upp	upper 95% (Wald) confidence bound

The confidence bounds assume that the function f is a negative log-likelihood

Note

newton computes the (logit-transformed) Hessian of f (using logit.hessian). This function is part of the Bhat exploration tool

Author(s)

E. Georg Luebeck (FHCRC)

See Also

dfp, ftrf, btrf, logit.hessian, plkhci

Examples

```
        # generate some Poisson counts on the fly
          dose <- c(rep(0,100),rep(1,100),rep(5,100),rep(10,100))
          data <- cbind(dose,rpois(400,20*(1+dose*.5*(1-dose*0.05))))

        # neg. log-likelihood of Poisson model with 'linear-quadratic' mean:
          lkh <- function (x) {
          ds <- data[, 1]
          y  <- data[, 2]
          g <- x[1] * (1 + ds * x[2] * (1 - x[3] * ds))
          return(sum(g - y * log(g)))
          }

# for example define
        x <- list(label=c("a","b","c"),est=c(10.,10.,.01),low=c(0,0,0),upp=c(100,20,.1))

# calls:
  r <- dfp(x,f=lkh)
```

```
x$est <- r$est
results <- newton(x,lkh)
```

---

plkhci *Profile-likelihood based confidence intervals*

Description

function to find prob*100% confidence intervals using profile-likelihood. Numerical solutions are obtained via a modified Newton-Raphson algorithm. The method is described in Venzon and Moolgavkar, Journal of the Royal Statistical Society, Series C vol 37, no.1, 1988, pp. 87-94.

Usage

```
plkhci(x, nlogf, label, prob = 0.95, eps = 0.001, nmax = 10, nfcn = 0)
```

Arguments

x	a list with components 'label' (of mode character), 'est' (the parameter vector with the initial guess), 'low' (vector with lower bounds), and 'upp' (vector with upper bounds)
nlogf	the negative log of the density function (not necessarily normalized) for which samples are to be obtained
label	parameter for which confidence bounds are computed
prob	probability associated with the confidence interval
eps	a numerical value. Convergence results when all (logit-transformed) derivatives are smaller eps
nmax	maximum number of Newton-Raphson iterations in each direction
nfcn	number of function calls

Value

2 component vector giving lower and upper p% confidence bounds

Note

At this point, only a single parameter label can be passed to plkhci. This function is part of the Bhat exploration tool

Author(s)

E. Georg Luebeck (FHCRC)

See Also

dfp, newton, logit.hessian

Examples

```
          # generate some Poisson counts on the fly
            dose <- c(rep(0,50),rep(1,50),rep(5,50),rep(10,50))
            data <- cbind(dose,rpois(200,20*(1+dose*.5*(1-dose*0.05))))

          # neg. log-likelihood of Poisson model with 'linear-quadratic' mean:
            nlogf <- function (x) {
            ds <- data[, 1]
            y  <- data[, 2]
            g <- x[1] * (1 + ds * x[2] * (1 - x[3] * ds))
            return(sum(g - y * log(g)))
            }

  # for example define
          x <- list(label=c("a","b","c"),est=c(10.,10.,.01),low=c(0,0,0),upp=c(100,20,.1))

  # get MLEs using dfp:
    r <- dfp(x,f=nlogf)
            x$est <- r$est
            plkhci(x,nlogf,"a")
            plkhci(x,nlogf,"b")
            plkhci(x,nlogf,"c")
  # e.g. 90% confidence bounds for "c"
            plkhci(x,nlogf,"c",prob=0.9)
```

Index