Package ‘bvls’

July 22, 2025

Type Package

Title The Stark-Parker algorithm for bounded-variable least squares

Version 1.4

Author Katharine M. Mullen

Maintainer Katharine M. Mullen <katharine.mullen@stat.ucla.edu>

Description An R interface to the Stark-Parker implementation of an algorithm for bounded-variable least squares

License GPL (>= 2)

Repository CRAN

Date/Publication 2013-12-18 08:53:26

NeedsCompilation yes

Contents

`bvls-package` *The Stark-Parker algorithm for bounded-variable least squares*

Description

An R interface to the Stark-Parker implementation of an algorithm for bounded-variable least squares that solves $\min \parallel Ax - b \parallel_2$ with the constraint $l \leq x \leq u$, where $l, x, u \in R^n, b \in R^m$ and A is an $m \times n$ matrix.

References

Stark PB, Parker RL (1995). Bounded-variable least-squares: an algorithm and applications, Computational Statistics, 10, 129-141.

See Also

bvls, the method "L-BFGS-B" for optim, solve.QP, nnls

`bvls` *The Stark-Parker implementation of bounded-variable least squares*

Description

An R interface to the Stark-Parker implementation of bounded-variable least squares that solves the least squares problem $\min \parallel Ax - b \parallel_2$ with the constraint $l \leq x \leq u$, where $l, x, u \in R^n, b \in R^m$ and A is an $m \times n$ matrix.

Usage

```
bvls(A, b, bl, bu, key=0, istate=rep(0,ncol(A)+1))
```

Arguments

`A`	numeric matrix with `m` rows and `n` columns
`b`	numeric vector of length `m`
`bl`	numeric vector of length `n` specifying the lower bound on each element of `x`
`bu`	numeric vector of length `n` specifying the upper bound on each element of `x`
`key`	If `key > 0` the routine initializes using the user's guess about which components of `x` are active, i.e. are strictly within their bounds, which are at their lower bounds, and which are at their upper bounds. This information is supplied through the array `istate`.
`istate`	vector of length `ncol(A)+1`. If `key > 0`, `istate` is as follows: the last contains the total number of components at their bounds (the bound variables). The absolute values of the first `nbound <- tail(istate,1)` entries of `istate` are the indices of these bound components of `x`. The sign of `istate[1:nbound]` indicates whether `x(abs(istate[1:nbound]))` is at its upper or lower bound. `istate[1:nbound]` is positive if the component is at its upper bound, negative if the component is at its lower bound. `istate[(nbound+1):ncol(A)]` contain the indices of the components of `x` that are active (i.e. are expected to lie strictly within their bounds). When `key > 0`, the routine initially sets the active components to the averages of their upper and lower bounds.

Value

`bvls` returns an object of class `"bvls"`.

The generic assessor functions `coefficients`, `fitted.values`, `deviance` and `residuals` extract various useful features of the value returned by `bvls`.

An object of class `"bvls"` is a list containing the following components:

`x`	the parameter estimates.

`deviance`	the residual sum-of-squares.
`residuals`	the residuals, that is response minus fitted values.
`fitted`	the fitted values.

Source

This is an R interface to the Fortran77 code accompanying the article referenced below by Stark PB, Parker RL (1995), and distributed via the **statlib** on-line software repository at Carnegie Mellon University (URL `http://lib.stat.cmu.edu/general/bvls`). The code was modified slightly to allow compatibility with the gfortran compiler. The authors have agreed to distribution under GPL version 2 or newer.

References

Stark PB, Parker RL (1995). Bounded-variable least-squares: an algorithm and applications, Computational Statistics, 10, 129-141.

See Also

the method `"L-BFGS-B"` for optim, solve.QP, nnls

Examples

```
## simulate a matrix A
## with 3 columns, each containing an exponential decay
t <- seq(0, 2, by = .04)
k <- c(.5, .6, 1)
A <- matrix(nrow = 51, ncol = 3)
Acolfunc <- function(k, t) exp(-k*t)
for(i in 1:3) A[,i] <- Acolfunc(k[i],t)

## simulate a matrix X
X <- matrix(nrow = 50, ncol = 3)
wavenum <- seq(18000,28000, length=nrow(X))
location <- c(25000, 22000)
delta <- c(1000,1000)
Xcolfunc <- function(wavenum, location, delta)
  exp( - log(2) * (2 * (wavenum - location)/delta)^2)
for(i in 1:2) X[,i] <- Xcolfunc(wavenum, location[i], delta[i])

X[1:40,3] <- Xcolfunc(wavenum, 23000, 1000)[11:nrow(X)]
X[41:nrow(X),3]<- - Xcolfunc(wavenum, 23000, 1000)[21:30]

## set seed for reproducibility
set.seed(3300)

## simulated data is the product of A and X with some
## spherical Gaussian noise added
matdat <- A %*% t(X) + .005 * rnorm(nrow(A) * nrow(X))

## estimate the rows of X using BVLS criteria
```

```
bvls_sol <- function(matdat, A) {
  X <- matrix(0, nrow = ncol(matdat), ncol = ncol(A) )
  bu <- c(Inf,Inf,.75)
  bl <- c(0,0,-.75)
  for(i in 1:ncol(matdat))
     X[i,] <- coef(bvls(A,matdat[,i], bl, bu))
  X
}
X_bvls <- bvls_sol(matdat,A)

matplot(X,type="p",pch=20)
matplot(X_bvls,type="l",pch=20,add=TRUE)
legend(10, -.5,
c("bound <= zero", "bound <= zero", "bound <= -.75 <= .75"),
col = c(1,2,3), lty=c(1,2,3),
text.col = "blue")

## Not run:
## can solve the same problem with L-BFGS-B algorithm
## but need starting values for x
bfgs_sol <- function(matdat, A) {
  startval <- rep(0, ncol(A))
  fn1 <- function(par1, b, A) sum( ( b - A %*% par1)^2)
  X <- matrix(0, nrow = ncol(matdat), ncol = 3)
  bu <- c(1000,1000,.75)
  bl <- c(0,0,-.75)
  for(i in 1:ncol(matdat))
    X[i,] <-  optim(startval, fn = fn1, b=matdat[,i], A=A,
                  upper = bu, lower = bl,
                  method="L-BFGS-B")$par
    X
}
X_bfgs <- bfgs_sol(matdat,A)

## the RMS deviation under BVLS is less than under L-BFGS-B
sqrt(sum((X - X_bvls)^2)) < sqrt(sum((X - X_bfgs)^2))

## and L-BFGS-B is much slower
system.time(bvls_sol(matdat,A))
system.time(bfgs_sol(matdat,A))

## End(Not run)
```

Index