Optimizing abyss with optimx

Hamid Younesy

October 16, 2016

"optimx" is a general-purpose optimization wrapper function that calls other R tools for optimization.

Usage

```
optimx(par, # a vector of initial values for the parameters for which optimal values are to be found.
       fn,  # A function to be minimized, with first argument the vector of parameters
       method=c("Nelder-Mead","BFGS"),  # list of the methods to be used (can be more than one)
       lower=-Inf, upper=Inf, # Bounds on the variables
       itnmax=NULL, # maximum number of iterations
       control=list(),
       ...)
```

Possible method codes are: 'Nelder-Mead', 'BFGS', 'CG', 'L-BFGS-B', 'nlm', 'nlminb', 'spg', 'ucminf', 'newuoa', 'bobyqa', 'nmkb', 'hjkb', 'Rcgmin', or 'Rvmmin'.

Here is an example of trying optix for all methods

```
library(optimx)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

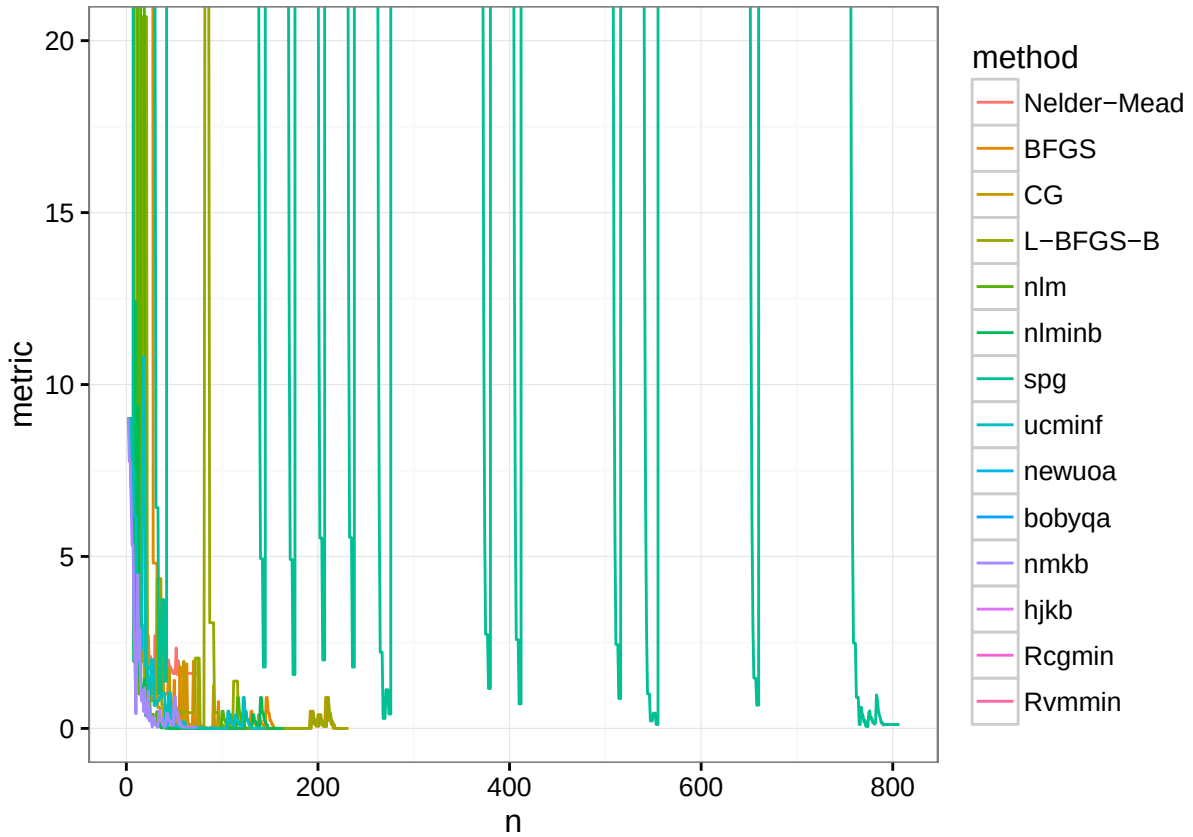
```
testOptimx <- function(func, init_param) {
    df_all <<- NULL
    ans_all <<- NULL
    for (mtd in c('Nelder-Mead' , 'BFGS', 'CG', 'L-BFGS-B'
                  , 'nlm', 'nlminb', 'spg', 'ucminf', 'newuoa'
                  , 'bobyqa', 'nmkb', 'hjkb', 'Rcgmin', 'Rvmmin'
                  )) {
        results <<- NULL
        ncall <<- 1
        ans <- optimx(
            fn = function(x) {
                r <- func(x)
                results <<- rbind(results, c(ncall, r, x))
                ncall <<- ncall + 1
                r
                }
            , method = mtd,
            #, lower=-20, upper=20
            , par = init_param
            , itnmax = 30
            #, control=list(all.methods=TRUE, save.failures=TRUE, trace=0)
            )
        df_all <<- rbind(df_all, cbind(data.frame(results), mtd))
```

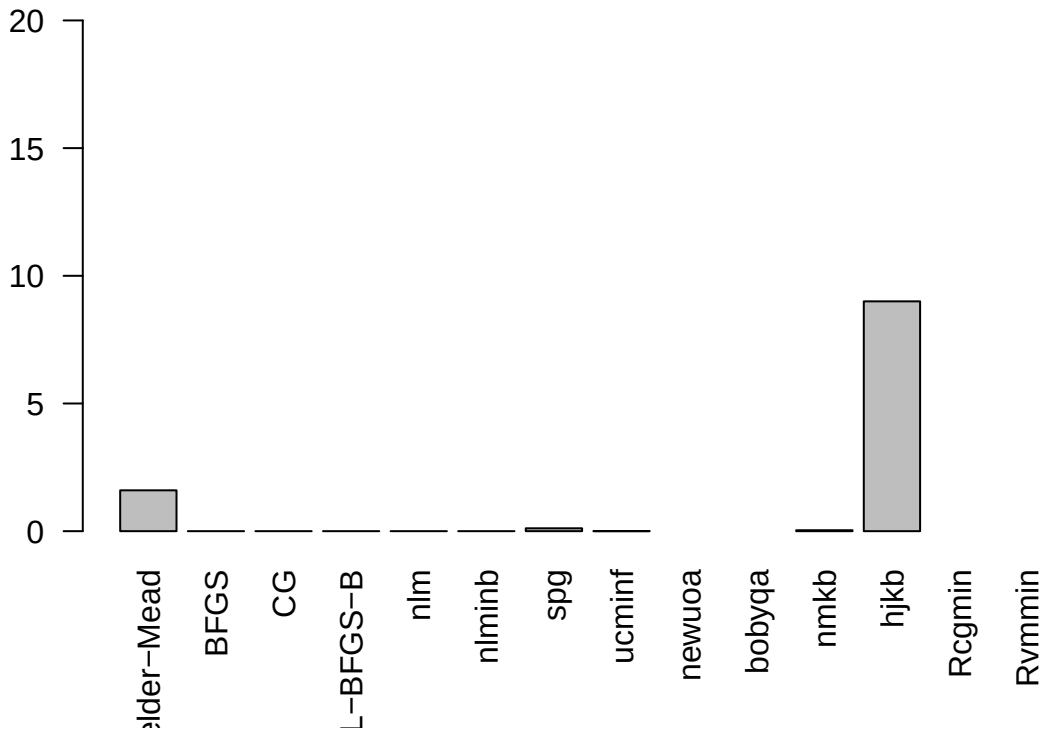
```
        ans_all <<- rbind(ans_all, ans)
    }

    colnames(df_all) <- c("n", "metric", paste("p",1:length(init_param)), "method")

    p <- ggplot(df_all, aes(color=method)) +
        theme_bw() +
        geom_line(aes(x=n, y=metric)) +
        coord_cartesian(ylim = c(0, 20))

    print(p)

    barplot(ans_all$value, names=rownames(ans_all), ylim=c(0,20), las=2)

    #print(ans_all)
}
```

Two continious parameters

```
testOptimx(
    func = function(x){abs(x[1]-5) + abs(x[2]-4)}
    , init_param = c(0, 0)
    )
```

```
## step nofc    fmin    xpar
```

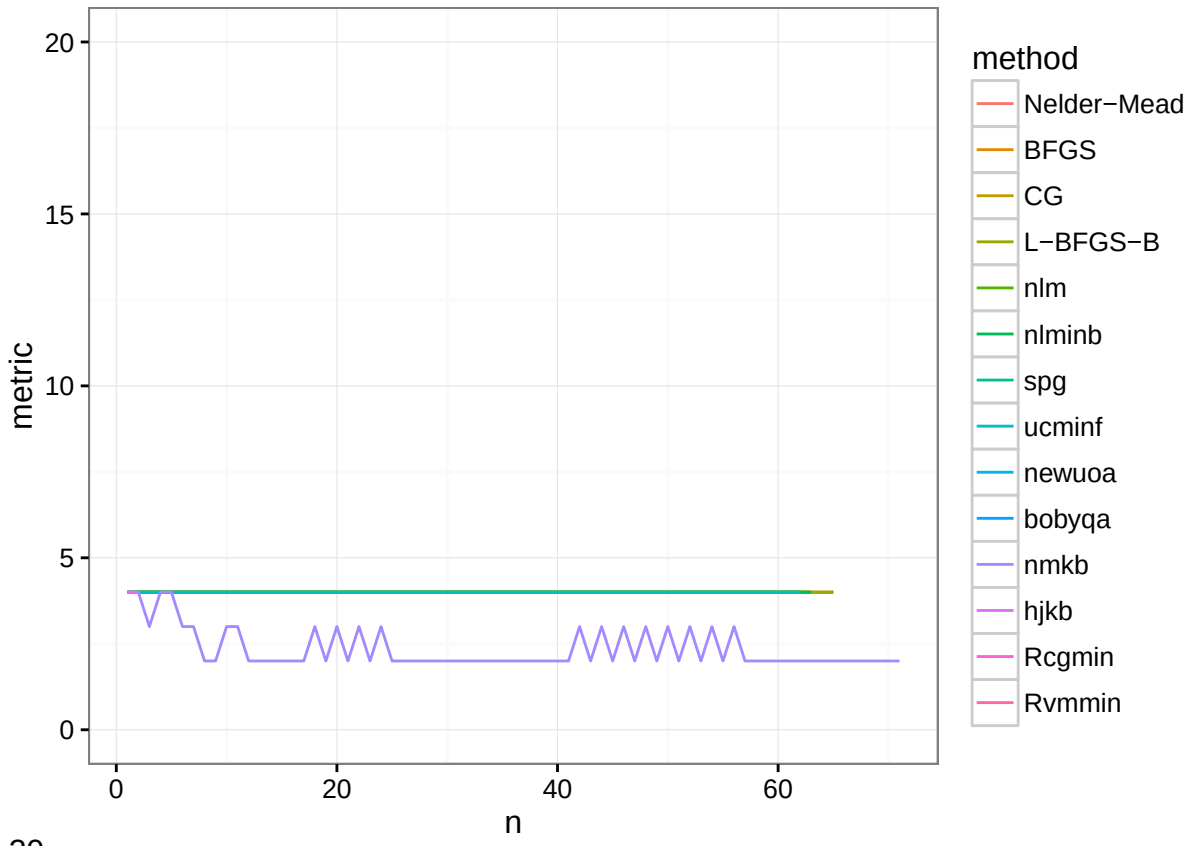


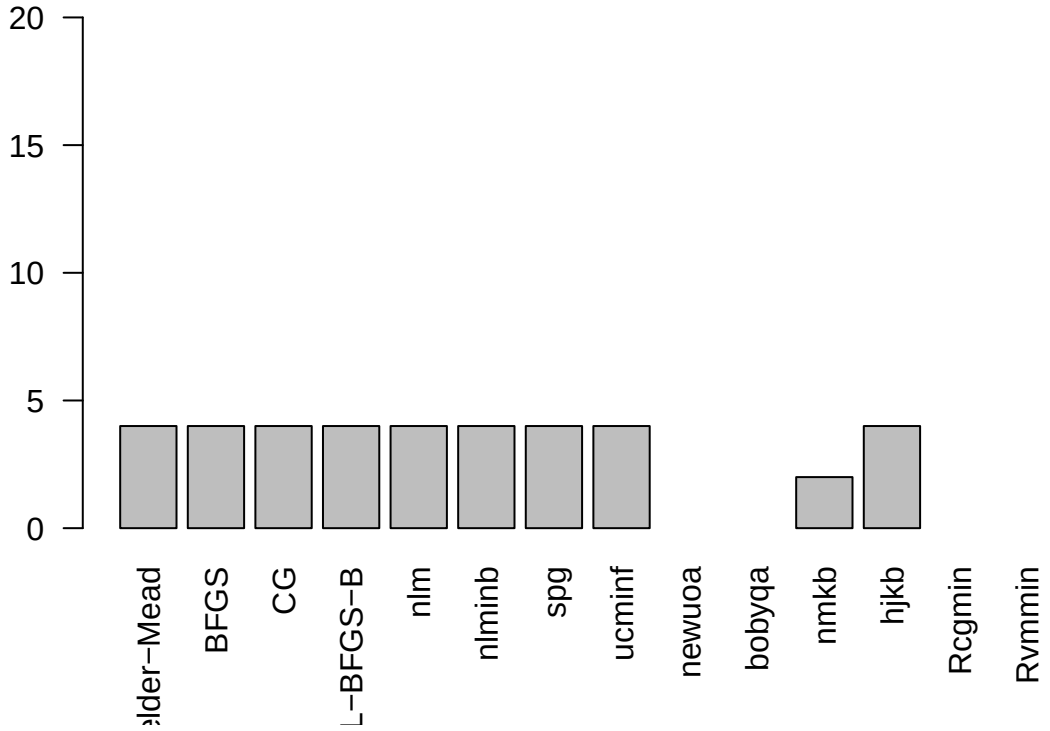


one discrete parameter

```
testOptimx(
    func = function(x){abs(round(x[1])-4)}
    , init_param = c(0,0,0)
    )
```

```
## step nofc    fmin    xpar
```

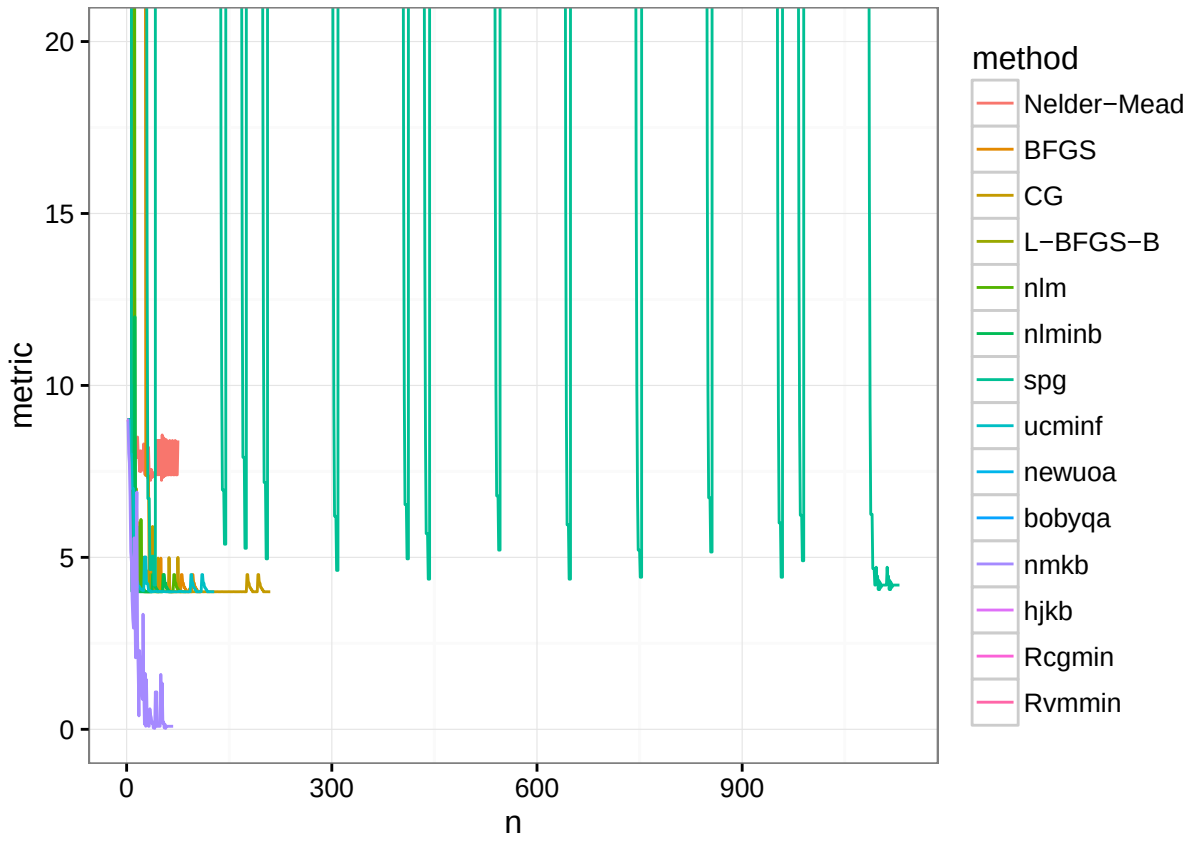

metric
n
method
Nelder−Mead
BFGS
CG
L−BFGS−B
nlm
nlminb
spg
ucminf
newuoa
bobyqa
nmkb
hjkb
Rcgmin
Rvmmin

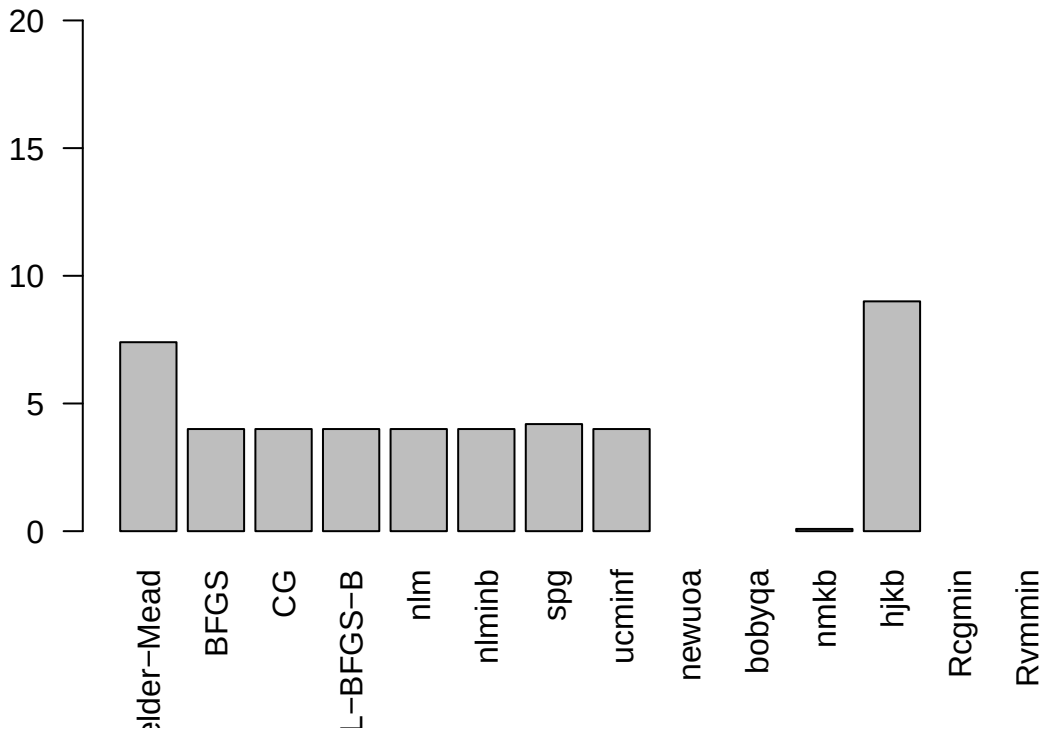

BFGS
CG
nlm
nlminb
spg
ucminf
newuoa
bobyqa
nmkb
hjkb
Rcgmin
Rvmmin

One continious and one discrete parameter

```
testOptimx(
    func = function(x){abs(x[1]-5) + abs(round(x[2])-4)}
    , init_param = c(0, 0)
    )
```

```
## step nofc    fmin    xpar
```

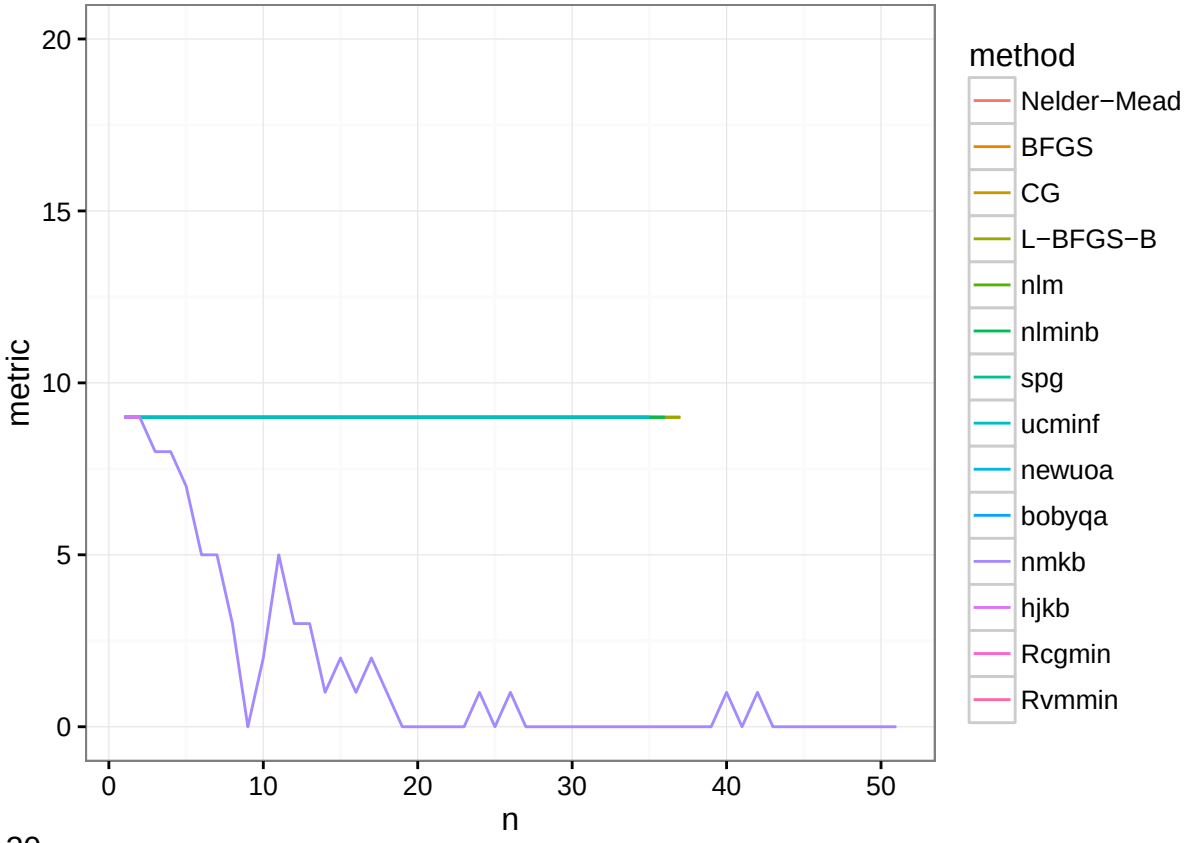


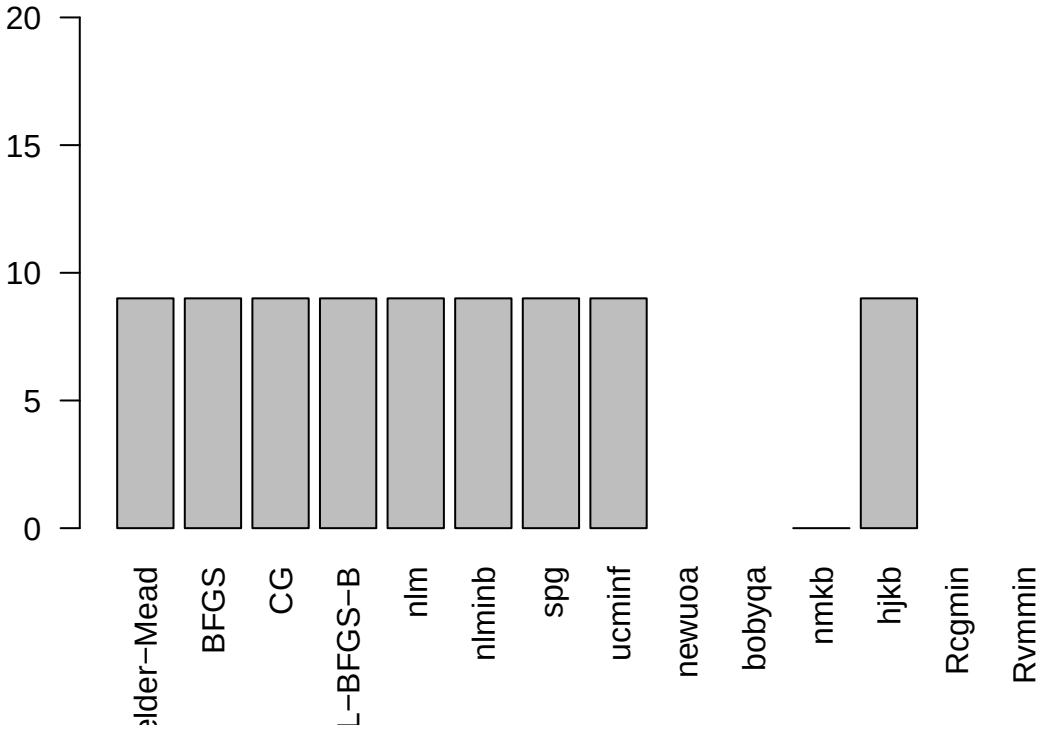


Two discrete parameters

```
testOptimx(
    func = function(x){abs(round(x[1])-5) + abs(round(x[2])-4)}
    , init_param = c(0, 0)
    )
```

```
## step nofc    fmin    xpar
```


metric
n
method
Nelder−Mead
BFGS
CG
L−BFGS−B
nlm
nlminb
spg
ucminf
newuoa
bobyqa
nmkb
hjkb
Rcgmin
Rvmmin


BFGS
CG
nlm
nlminb
spg
ucminf
newuoa
bobyqa
nmkb
hjkb
Rcgmin
Rvmmin