Faster CLHS

Kiri Daust

31/08/2020

Due to our frequency of use of the clhs function and its relative lack of speed, I have translated the function into C++ using the excellent Rcpp API. This has resulted in a speedup of ~ 100x, so running a clhs with 10000 iterations usually takes < 1 second.

How to Use

All the C++ code is contained in the file "CppLHS.cpp". The main function in the file (and the only one exported into R) is called CppLHS. This function runs the metropolis-hasting algorithm, and can be used by itself, but due to some awkardness with certain inputs, I've created an R wrapper to make the function call a bit nicer. The R wrapper is in the file FastCLHS_R.R, and if you source this file it will also source the C++ script. The function is called c_clhs and has the required parameters x (matrix of data), i_cost (index in matrix of cost values, NULL if not cost constrained), size (# of required samples). There are also optional parameters: include (indices of already sampled data), iter (number of metropolis-hasting interations), temp (initial temperature), tdecrease (decrease in metropolis temperature), eta (number of expected samples in each bin), and length.cycle(how long between temperature decreases).

The output from my adaptation also varies slightly from the original. The output is a list which contains + sampled_data (matrix of data) + indeces (indices of sampled data - sorry for the typo) + obj (objective values for each iteration) + final_obj (objective values for each sample in the final iteration - lower values mean closer to a perfect latin hypercube)

Example

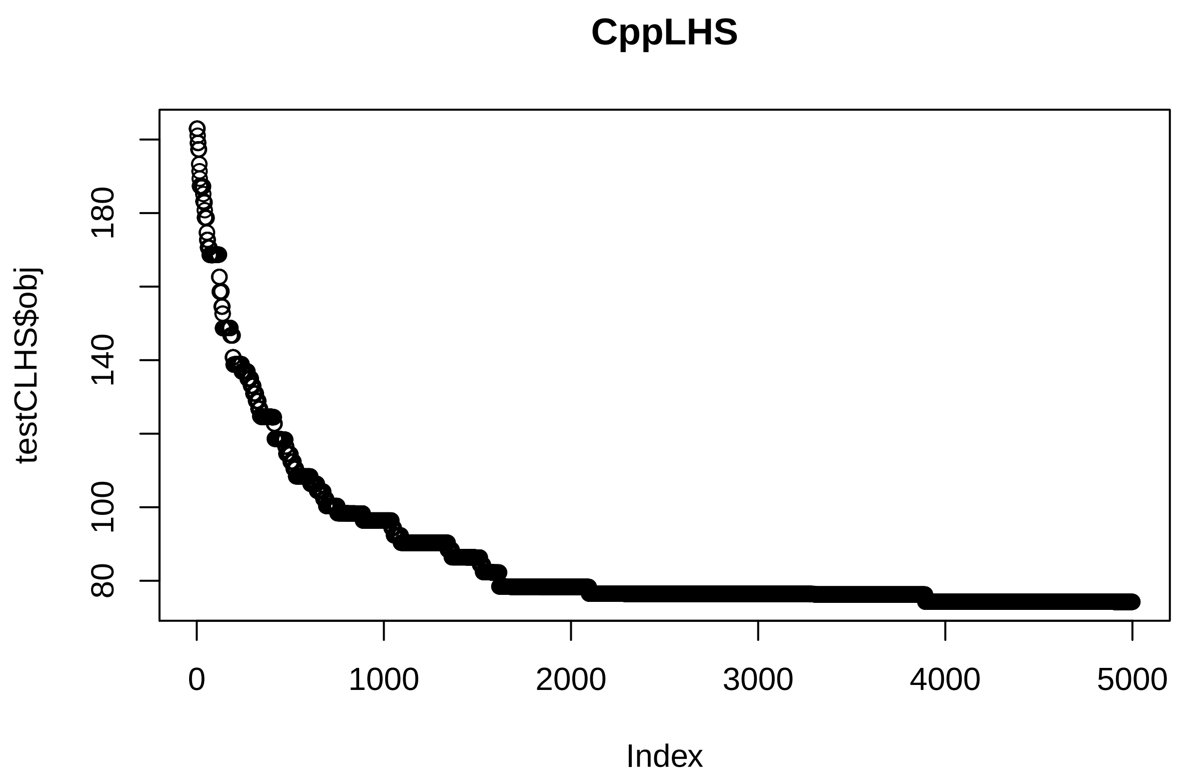
```r
##create sample data
nsamp = 100000
df <- data.frame(
  a = runif(nsamp),
  b = rnorm(nsamp),
  c = rexp(nsamp)*4,
  d = rgamma(nsamp, shape = 2),
  cost = runif(nsamp, min = 0, max = 5)
)
nsample = 67
df <- as.matrix(df)

source("FastCLHS_R.R")

tic()
testCLHS <- c_clhs(df, nsample, include = NULL, i_cost = 5, iter = 5000)
```

```
## vroom vroom
```

```r
toc()
```

```
## 0.264 sec elapsed
```

```r
plot(testCLHS$obj, main = "CppLHS")
```



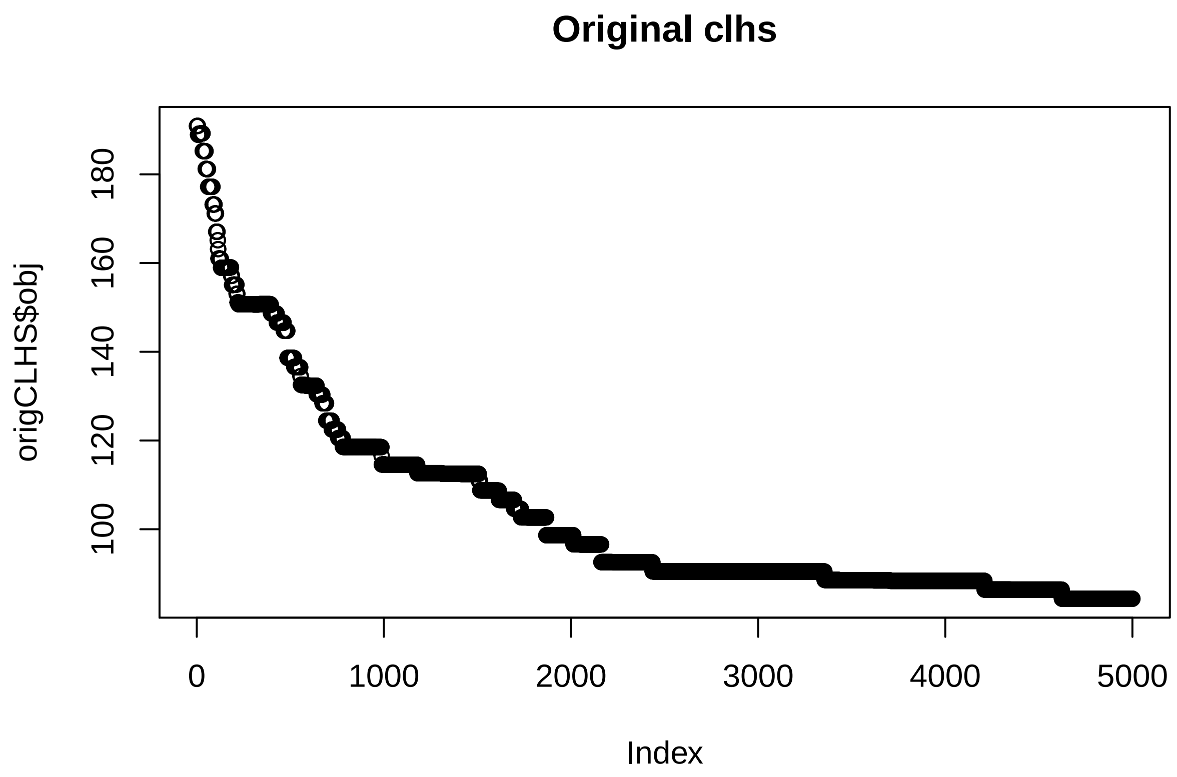
```r
tic()
origCLHS <- clhs(as.data.frame(df), nsample, cost = 5, iter = 5000,simple = F)
```

```
##   |                                                                                |
```

```r
toc()
```

```
## 42.13 sec elapsed
```

```r
plot(origCLHS$obj, main = "Original clhs")
```



Note on parallelisation

To use this function in parallel, it's necessary to compile the C++ code on all threads. This can be done as follows.

```
worker.init <- function(){
  Rcpp::sourceCpp("CppCLHS.cpp")
}

require(doParallel)
cl <- makePSOCKcluster(detectCores()-2)
clusterCall(cl, worker.init)
registerDoParallel(cl)


###eg
foreach(n = nums, .combine = rbind, .noexport = c("CppLHS"),
        .export = c("c_clhs")) %dopar% {
          xxx
        }
```