PM566 Lab 9

Echo Tang

2022-10-26

Problem 1

Three problems that can be solved by parallel computing are: 1. Cross-validation for testing machine learning models (i.e. structure predictions for macromolecules) 2. Resampling-based testing for very large datasets for hypothesis testing 3. Text mining for large datasets

Problem 2

Part 1

```
library(parallel)
library(microbenchmark)

fun1 <- function(n = 100, k = 4, lambda = 4) {
  x <- NULL

  for (i in 1:n)
    x <- rbind(x, rpois(k, lambda))

  return(x)
}

fun1alt <- function(n = 100, k = 4, lambda = 4) {
  x = matrix(rpois(n*k, lambda), ncol = k)
  return(x)

}

microbenchmark::microbenchmark(
  fun1(),
  fun1alt()
)
```

```
## Unit: microseconds
##       expr     min       lq      mean   median      uq       max neval
##     fun1() 296.876 327.3135 550.25273 342.9175 355.355 21313.709   100
##  fun1alt()  18.667  19.8545  39.16522  20.9380  22.501  1765.959   100
```

Part 2

```r
set.seed(1234)
M = matrix(runif(12), ncol = 4)

# Find each column's max value
fun2 <- function(x) {
  apply(x, 2, max)
}
fun2(M)
```

```
## [1] 0.6222994 0.8609154 0.6660838 0.6935913
```

```r
fun2alt <- function(x) {
  idx = max.col(t(x))
  x[cbind(idx,1:4)]
}
fun2alt(M)
```

```
## [1] 0.6222994 0.8609154 0.6660838 0.6935913
```

```r
# Benchmarking
microbenchmark::microbenchmark(
  fun2(M),
  fun2alt(M)
)
```

```
## Unit: microseconds
##        expr    min      lq     mean median      uq      max neval
##     fun2(M) 16.292 17.3550 27.77399 17.938 20.2510  936.125   100
##  fun2alt(M) 12.500 12.8965 36.80780 13.584 14.7715 2298.834   100
```

Problem 3

Part 1

```r
my_boot <- function(dat, stat, R, ncpus = 1L) {

  n <- nrow(dat)
  idx <- matrix(sample.int(n, n*R, TRUE), nrow=n, ncol=R)

  # Making the cluster using `ncpus`
  cl = makePSOCKcluster(ncpus)
  clusterSetRNGStream(cl, 123)
  clusterExport(cl, c("stat", "dat", "idx"), envir = environment())

  ans <- parLapply(cl, seq_len(R), function(i) {
    stat(dat[idx[,i], , drop=FALSE])
  })
```

```r
  # Coercing the list into a matrix
  ans <- do.call(rbind, ans)

  # STEP 4: GOES HERE

  ans

}
```

Part 2

```r
my_stat <- function(d) coef(lm(y ~ x, data=d))

set.seed(1)
n <- 500; R <- 1e4

x <- cbind(rnorm(n)); y <- x*5 + rnorm(n)

ans0 <- confint(lm(y~x))
ans1 <- my_boot(dat = data.frame(x, y), my_stat, R = R, ncpus = 2L)
t(apply(ans1, 2, quantile, c(.025,.975)))
```

```
##                   2.5%      97.5%
## (Intercept) -0.1386903 0.04856752
## x            4.8685162 5.04351239
```

```r
ans0
```

```
##                  2.5 %     97.5 %
## (Intercept) -0.1379033 0.04797344
## x            4.8650100 5.04883353
```

```r
system.time(my_boot(dat = data.frame(x, y), my_stat, R = 4000, ncpus = 1L))
```

```
##    user  system elapsed
##   0.067   0.013   2.399
```

```r
system.time(my_boot(dat = data.frame(x, y), my_stat, R = 4000, ncpus = 2L))
```

```
##    user  system elapsed
##   0.089   0.017   1.437
```