`mc.cores`Statistics 506

During lecture, we encountered a seeming contradiction, where manually setting the `mc.cores` argument to `parallel::mclapply` resulted in better performance than the default, which I had thought was the value of `detectCores()`.

The default argument to `mc.cores` is not the value of `detectCores()`, but is instead 2.

Additionally, we saw that we could increase the argument to `mc.cores` above `detectCores()`, it turns out that `mclapply` isn't "core-aware" - it simply spawns new processes, so you could set `mc.cores` as high you'd like, but once you get above the number of available cores, it will likely slow you down even more as a single core needs to switch between processes.

Here's a demonstration.

```r
library(parallel)
detectCores()
```

```
[1] 8
```

The machine I am compiling these notes on has access to 8 cores.

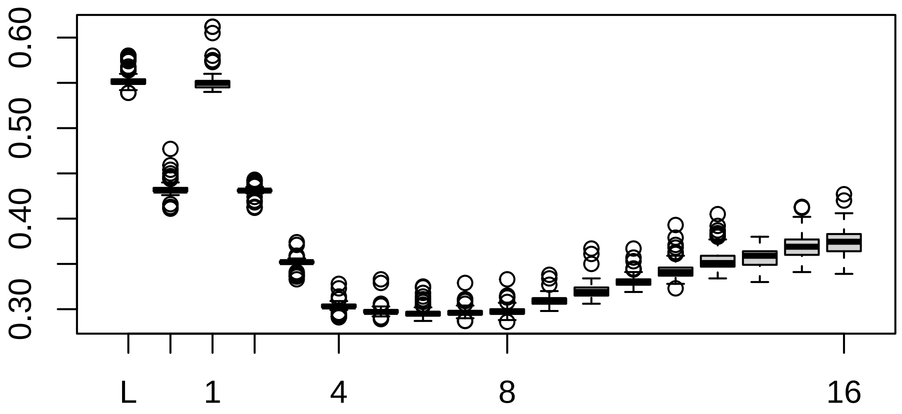
```r
library(lme4)
```

```
Loading required package: Matrix
```

```r
f <- function(dat) {
     lmer(Petal.Width ~ . - Species + (1 | Species),
          data = dat)
}
```

We'll run a simulation for `mc.cores = 1` through `mc.cores = 16`, as well as leaving it at it's default of 2.

```r
reps <- 100
savemat <- matrix(rep(-1, reps*18), ncol = 18)

for (i in seq_len(reps)) {
  # lapply
  savemat[i, 1] <-
    system.time(lapply(1:100, function(x) f(iris)))["elapsed"]

  # mclapply with the default argument
  savemat[i, 2] <-
    system.time(mclapply(1:100, function(x) f(iris)))["elapsed"]

  # mclapply with increasing `mc.cores` argument
  for (j in 1:16) {
    savemat[i, j + 2] <-
      system.time(mclapply(1:100,
                           function(x) f(iris),
                           mc.cores = j))["elapsed"]
  }
}

# remove any outliers that will make the plot look bad:

savemat <- apply(savemat, 2, function(x) {
  x[x > mean(x) + 3*sd(x) | x < mean(x) - 3*sd(x)] <- NA
  return(x)
})

boxplot(savemat, xaxt = "n")
# Add a nicer axis - "L" for lapply, "D" for default
axis(1, at = c(1, 2, 3, 4, 6, 10, 18),
     labels = c("L", "D", 1, 2, 4, 8, 16))
```



So we see

1. `mclapply` with `mc.cores = 1` simply calls `lapply`, so performance is similar.
2. Leaving `mc.cores` at its default does indeed give performance equivalent to `mc.cores =2`.
3. As we move above the 8 cores my machine has, peformance continues to degrade.