Assignment 2: Parallelisation

Mishka Ramraj

Table of contents

Chapter 1

Assignment 2: Parallelisation

1.0.1 Requirements

My Github respository link: https://github.com/Mishka312/StatHonPrac2/tree/main

Load necessary packages:

```
library(doParallel)
library(foreach)
library(MASS)
library(boot)
library(iterators)
library(knitr)
```

1.1 Question 1

Below is the code associated with 2 methods. The first uses foreach() for a sequential functionality and the second for parallelisation.

```
exp_random <- function() {
  sample <- rexp(1000, 1)
  m <- mean(sample)
  v <- var(sample)

  vals <- list(mean = m, variance = v)
```

```
}


## Sequentially
results1.1 <- foreach(i = 1:100, .combine = rbind) %do% {
  exp_random()
}


## Parallelisation

cl <- makeCluster(detectCores() - 1)
registerDoParallel(cl)

results1.2 <- foreach(i = 1:100, .combine = rbind) %dopar% {
  exp_random()
}

stopCluster(cl)
```

1.2 Question 2

Table 1.1 displays that sequential methods are quicker when just considering one bootstrap sample at a time. However, when considering 100 Bootstraps samples at a time, parallelisation yields a shorter time. In this demonstration, parallelisation using multiple bootstraps samples at once was faster by about 2.7 seconds.

Table 1.1: Time taken to perform bootstrapping tasks using sequential and parallelised methods.

	user.self	sys.self	elapsed	user.child	sys.child
single_parallel	0.07	0.05	0.55	NA	NA
single_sequential	0.00	0.00	0.02	NA	NA
multiple_parallel	0.08	0.05	1.78	NA	NA
multiple_sequential	4.19	0.08	4.28	NA	NA

1.3 Question 3

In order to estimate the coverage of a bootstrap confidence interval, a sample was taken from the exponential(1) distribution. From this, the sample statistic

(mean) was calculated and 100 boostraps taken. A 95% confidence interval was constructed. This process was repeated 1000 times so that we accumulated 1000 bootstrap confidence intervals. From all of these, the proportion of confidence intervals containing the true mean (1) was calculated. This told us the *coverage* of the bootstrap confidence intervals.

In this scenario, the value for coverage is 0.925.

1.4 Question 4

Below is the code to find the maximum of 3 vectors of length 5 using foreach and an iterator object from irnorm().

The resultant maximum values are shown below

```
set.seed(1234)

iterator <- irnorm(n = 5, mean = 1, sd = 1)

results4.1 <- foreach(i = 1:3, .combine = cbind, .packages='iterators') %do% {
  print(max(nextElem(iterator)))
}
```

```
[1] 2.084441
[1] 1.506056
[1] 1.959494
```

1.5 Question 5

The following table depicts that the foreach used for sequential functionality and the replicate method were the fastest methods. Using parLapply worked in 0.4 seconds and the foreach using parallelisation took the longest - 0.47 seconds.

Table 1.2: Times taken for each method to find the maximum of a length-3 vector generated by irnorm()

	user.self	sys.self	elapsed	user.child	sys.child
foreach_do_time	0.00	0.00	0.00	NA	NA
foreach_dopar_time	0.04	0.03	0.45	NA	NA
replicate_time	0.00	0.00	0.01	NA	NA
parLapply_time	0.06	0.03	0.40	NA	NA