Parallel Computing Prac

STPLAR002

Question 1:

```r
output <- foreach(i = 1:100, .combine = rbind) %do% {
  data <- rexp(100, rate = 1)
  mean_data <- mean(data)
  var_data <- var(data)
  c(Mean = mean_data, Variance = var_data)
}
x <-  head(output)
```

The first 6 means and variances:

```
              Mean  Variance
result.1 1.0313472 0.7922345
result.2 0.9930388 1.2765253
result.3 0.9392942 0.7208715
result.4 1.0487265 1.0601242
result.5 0.9267843 0.6855783
result.6 1.0667263 0.8806277
```

Question 2:

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

runs <- 1000
size <- 1000
```

```r
boot_medians <- foreach(i = 1:runs, .combine = c, .packages = 'MASS') %dopar% {
    sample_data <- sample(galaxies, replace = TRUE)
    median(sample_data)
}

single_par_times <- system.time(
  boot_medians <- foreach(i = 1:runs, .combine = c, .packages = 'MASS') %dopar% {
    sample_data <- sample(galaxies, replace = TRUE)
    median(sample_data)
    }
)

serial_times <- system.time(
  serial_boot_meds <- replicate(runs, median(sample(galaxies, replace = TRUE)))
)


par_bs_batch_fn <- function(runs, size = 1000) {
  num_batches <- runs / size

  unlist(foreach(i = 1:num_batches, .combine = c, .packages = 'MASS') %dopar% {
    replicate(size, median(sample(galaxies, replace = TRUE)))
  })
}

batch_par_times <-  system.time(
  par_bs_medians_batch <- par_bs_batch_fn(runs, size)
)

stopCluster(cl)
```

Summary of the bootstrap medians from single sample:

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  19863   20522   20834   20874   21062   22374
```

Serial Processing Time:

```
   user  system elapsed
   0.06    0.00    0.07
```

Parallel (Single Sample) Time:

```
user  system elapsed
0.43    0.08    0.50
```

Parallel (Batching 1000 Samples) Time:

```
user  system elapsed
0.02    0.00    0.06
```

Question 3:

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

true_mean <- 1

coverage_est <- function() {
  sample_data <- rexp(50, rate = 1)

  boot_means <- replicate(1000, mean(sample(sample_data, replace = TRUE)))

  CI_lower <- quantile(boot_means, 0.025)
  CI_upper <- quantile(boot_means, 0.975)
  coverage <- as.numeric(CI_lower <= 1 & CI_upper >= 1)
  return(coverage)
}

coverage_results <- foreach(i = 1:1000, .combine = c) %dopar% {
  coverage_est()
}

stopCluster(cl)

coverage_probability <- round(mean(coverage_results), 4)
```

Estimated Coverage Probability: 0.922

Question 4:

```
set.seed(1234)

vector_list <- list(irnorm(5), irnorm(5), irnorm(5))

max_value <- foreach(i = vector_list, .combine = c) %do% {
  max(nextElem(i))
}

find_max <- function() {
  it <- irnorm(5)  # Iterator for 5 normal random numbers
  vectors <- as.numeric(nextElem(it, 5))  # Convert iterator to a numeric vector
  return(max(vectors))  # Return the max value
}

foreach_result <- foreach(i = 1:3, .combine = c) %do% find_max()
print(foreach_result)
```

```
[1] 2.4158352 0.4595894 0.5747557
```

The largest values in the vectors:

Vector 1: 2.4158352

Vector 2: 0.4595894

Vector 3: 0.5747557

Question 5:

```
set.seed(1234)

# Function to generate 5 random numbers and return the max
find_max <- function() {
  it <- irnorm(5)  # Iterator for 5 normal random numbers
  vectors <- as.numeric(nextElem(it, 5))  # Convert iterator to a numeric vector
  return(max(vectors))  # Return the max value
}
```

```r
# 1. Using foreach (parallel)
system.time({
  foreach_result <- foreach(i = 1:3, .combine = c) %do% find_max()
  print(foreach_result)
})
```

```
[1] 1.0844412 0.5060559 0.9594941
```

```
   user  system elapsed 
      0       0       0 
```

```r
# 3. Using replicate (sequential)
system.time({
  replicate_result <- replicate(3, find_max())
  print(replicate_result)
})
```

```
[1] 2.4158352 0.4595894 0.5747557
```

```
   user  system elapsed 
   0.01    0.00    0.00 
```