W2-Assignment

Wenshan, Liu

2025-01-31

##### Q1. Q2

# load libraries  
  
library(MASS) # for data set   
library(parallel)   
library(doParallel)

## Warning: package 'doParallel' was built under R version 4.3.3

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 4.3.3

## Loading required package: iterators

library(foreach) # For parallel processing  
library(ggplot2) # For plotting  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':  
##   
## select

## The following objects are masked from 'package:stats':  
##   
## filter, lag

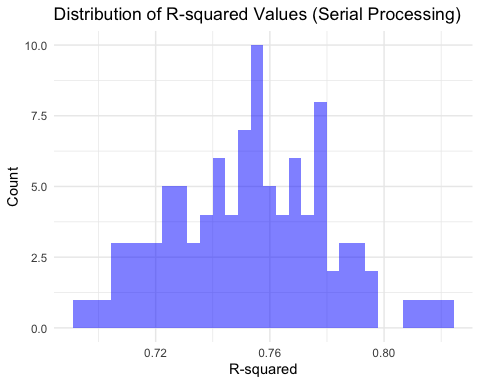
## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(glue)  
  
# Q1   
set.seed(42) # set seed = 42  
data("Boston")  
  
# Step 1: Generate 100 bootstrapped samples first  
n\_samples <- 100  
n\_rows <- nrow(Boston)  
bootstrap\_samples <- replicate(n\_samples,   
 sample(1:n\_rows, replace = TRUE),   
 simplify = FALSE)  
  
# Function to fit GLM model and extract R-squared  
fit\_glm\_and\_get\_rsquared <- function(indices) {  
 sample\_data <- Boston[indices, ]  
 model <- glm(medv ~ ., data = sample\_data, family = gaussian)  
 rsq <- with(summary(model), 1 - deviance/null.deviance)  
 return(rsq)  
}  
  
# Step 2: Serially fit models on each bootstrap sample  
start\_time <- Sys.time()  
serial\_results <- vector("numeric", n\_samples)  
  
for(i in 1:n\_samples) {  
 # Fit model on i-th bootstrap sample  
 serial\_results[i] <- fit\_glm\_and\_get\_rsquared(bootstrap\_samples[[i]])  
   
}  
  
end\_time <- Sys.time()  
serial\_duration <- difftime(end\_time, start\_time, units = "secs")  
  
  
# Print execution\_time   
print(paste("Execution time:" ,serial\_duration))

## [1] "Execution time: 0.140710115432739"

#### Q3

# Q3:Calc mean, IQR and plot the result   
  
# Calculate summary statistics for serial results  
serial\_stats <- list(  
 mean = mean(serial\_results),  
 q1 = quantile(serial\_results, 0.25),  
 q3 = quantile(serial\_results, 0.75),  
 iqr = IQR(serial\_results)  
)  
  
# Create plot for serial results  
serial\_plot <- ggplot(data.frame(R\_squared = serial\_results), aes(x = R\_squared)) +  
 geom\_histogram(bins = 30, fill = "blue", alpha = 0.5) +  
 theme\_minimal() +  
 labs(title = "Distribution of R-squared Values (Serial Processing)",  
 x = "R-squared",  
 y = "Count")  
  
print(serial\_plot)

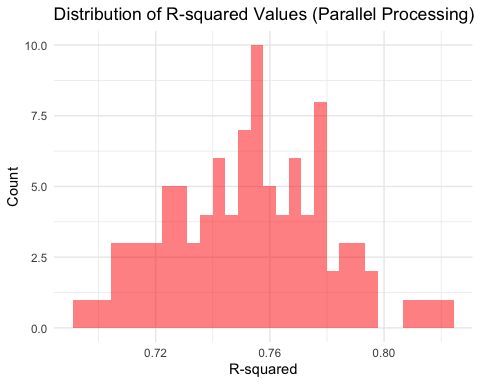


glue("  
Serial Processing Summary Statistics:  
Mean R-squared: {serial\_stats$mean}  
Q1: {serial\_stats$q1}  
Q3: {serial\_stats$q3}  
IQR: {serial\_stats$iqr}  
")

## Serial Processing Summary Statistics:  
## Mean R-squared: 0.752654635427502  
## Q1: 0.731018035522359  
## Q3: 0.772071895359253  
## IQR: 0.0410538598368941

##### Q4

#Q4   
  
# Set up parallel processing  
#num\_cores <- detectCores() # Use all available cores  
cl <- makeCluster(4)  
registerDoParallel(cl)  
  
# Parallel processing  
start\_time\_parallel <- Sys.time()  
  
parallel\_results <- foreach(indices = bootstrap\_samples,   
 .combine = c,  
 .packages = c("MASS")) %dopar% {  
 fit\_glm\_and\_get\_rsquared(indices)  
}  
  
  
end\_time\_parallel <- Sys.time()  
parallel\_duration <- difftime(end\_time\_parallel, start\_time\_parallel, units = "secs")  
  
stopCluster(cl)  
  
# Calculate summary statistics for parallel results  
parallel\_stats <- list(  
 mean = mean(parallel\_results),  
 q1 = quantile(parallel\_results, 0.25),  
 q3 = quantile(parallel\_results, 0.75),  
 iqr = IQR(parallel\_results)  
)  
# Create plot for Parallel results  
parallel\_plot <- ggplot(data.frame(R\_squared = parallel\_results), aes(x = R\_squared)) +  
 geom\_histogram(bins = 30, fill = "Red", alpha = 0.5) +  
 theme\_minimal() +  
 labs(title = "Distribution of R-squared Values (Parallel Processing)",  
 x = "R-squared",  
 y = "Count")  
  
print(parallel\_plot)

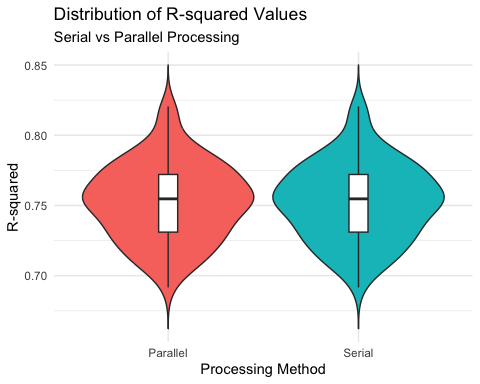


glue("  
Parallel Processing Summary Statistics:  
Mean R-squared: {parallel\_stats$mean}  
Q1: {parallel\_stats$q1}  
Q3: {parallel\_stats$q3}  
IQR: {parallel\_stats$iqr}  
")

## Parallel Processing Summary Statistics:  
## Mean R-squared: 0.752654635427502  
## Q1: 0.731018035522359  
## Q3: 0.772071895359253  
## IQR: 0.0410538598368941

##### Q5

# Q5  
# Create comparison plot  
plot\_data <- data.frame(  
 R\_squared = c(serial\_results, parallel\_results),  
 Method = rep(c("Serial", "Parallel"), each = n\_samples)  
)  
  
comparison\_plot <- ggplot(plot\_data, aes(x = Method, y = R\_squared, fill = Method)) +  
 geom\_violin(trim = FALSE) +  
 geom\_boxplot(width = 0.1, fill = "white") +  
 theme\_minimal() +  
 labs(title = "Distribution of R-squared Values",  
 subtitle = "Serial vs Parallel Processing",  
 y = "R-squared",  
 x = "Processing Method") +  
 theme(legend.position = "none")  
  
print(comparison\_plot)



cat("\n Serial Process Summary SDtatistics: \n")

##   
## Serial Process Summary SDtatistics:

cat("Mean R-squared:", serial\_stats$mean, "\n")

## Mean R-squared: 0.7526546

cat("Q1:", serial\_stats$q1, "\n")

## Q1: 0.731018

cat("Q3:", serial\_stats$q3, "\n")

## Q3: 0.7720719

cat("IQR:", serial\_stats$iqr, "\n")

## IQR: 0.04105386

# Print parallel statistics and timing comparison  
cat("\nParallel Processing Summary Statistics:\n")

##   
## Parallel Processing Summary Statistics:

cat("Mean R-squared:", parallel\_stats$mean, "\n")

## Mean R-squared: 0.7526546

cat("Q1:", parallel\_stats$q1, "\n")

## Q1: 0.731018

cat("Q3:", parallel\_stats$q3, "\n")

## Q3: 0.7720719

cat("IQR:", parallel\_stats$iqr, "\n")

## IQR: 0.04105386

cat("\nExecution Times:\n")

##   
## Execution Times:

cat("Serial Processing:", serial\_duration, "seconds\n")

## Serial Processing: 0.1407101 seconds

cat("Parallel Processing:", parallel\_duration, "seconds\n")

## Parallel Processing: 0.07080197 seconds

library(knitr)

## Warning: package 'knitr' was built under R version 4.3.3

# Create comparison data frame  
stats\_comparison <- data.frame(  
 Metric = c("Mean R-squared", "Q1", "Q3", "IQR", "Execution Time (sec) "),  
 Serial = c(serial\_stats$mean, serial\_stats$q1, serial\_stats$q3, serial\_stats$iqr,   
 as.numeric(serial\_duration)),  
 Parallel = c(parallel\_stats$mean, parallel\_stats$q1, parallel\_stats$q3, parallel\_stats$iqr,  
 as.numeric(parallel\_duration))  
)  
  
# Create table   
kable(stats\_comparison,   
 digits = 4,  
 caption = "Comparison of Serial and Parallel Processing")

Comparison of Serial and Parallel Processing

| Metric | Serial | Parallel |
| --- | --- | --- |
| Mean R-squared | 0.7527 | 0.7527 |
| Q1 | 0.7310 | 0.7310 |
| Q3 | 0.7721 | 0.7721 |
| IQR | 0.0411 | 0.0411 |
| Execution Time (sec) | 0.1407 | 0.0708 |

Based on the report and visualizations, the distribution of model fit statistics is similar. Because of using the same seed number and sample size for both methods. However, as we can see the execution time is different. The parallel processing method is significantly faster than serial processing. Here is the reason. The parallel processing method would use multiple CPU cores, but the the serial processing method only use single CPU core. Therefore, the results of this experiment demonstrate that parallel processing is more efficient than serial processing.