W2-Assignment

Wenshan, Liu

2025-01-31

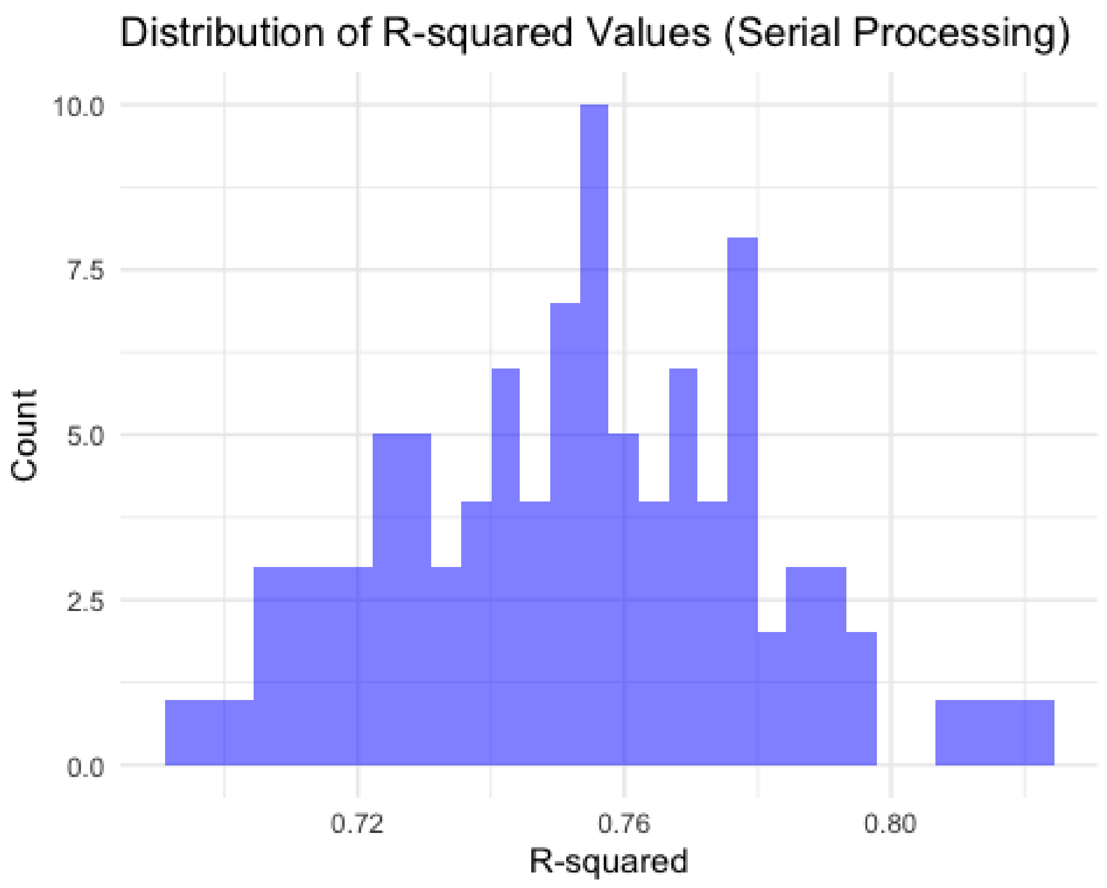
Q1. Q2

```
# 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)
```

```r
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

```r
# 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)
```



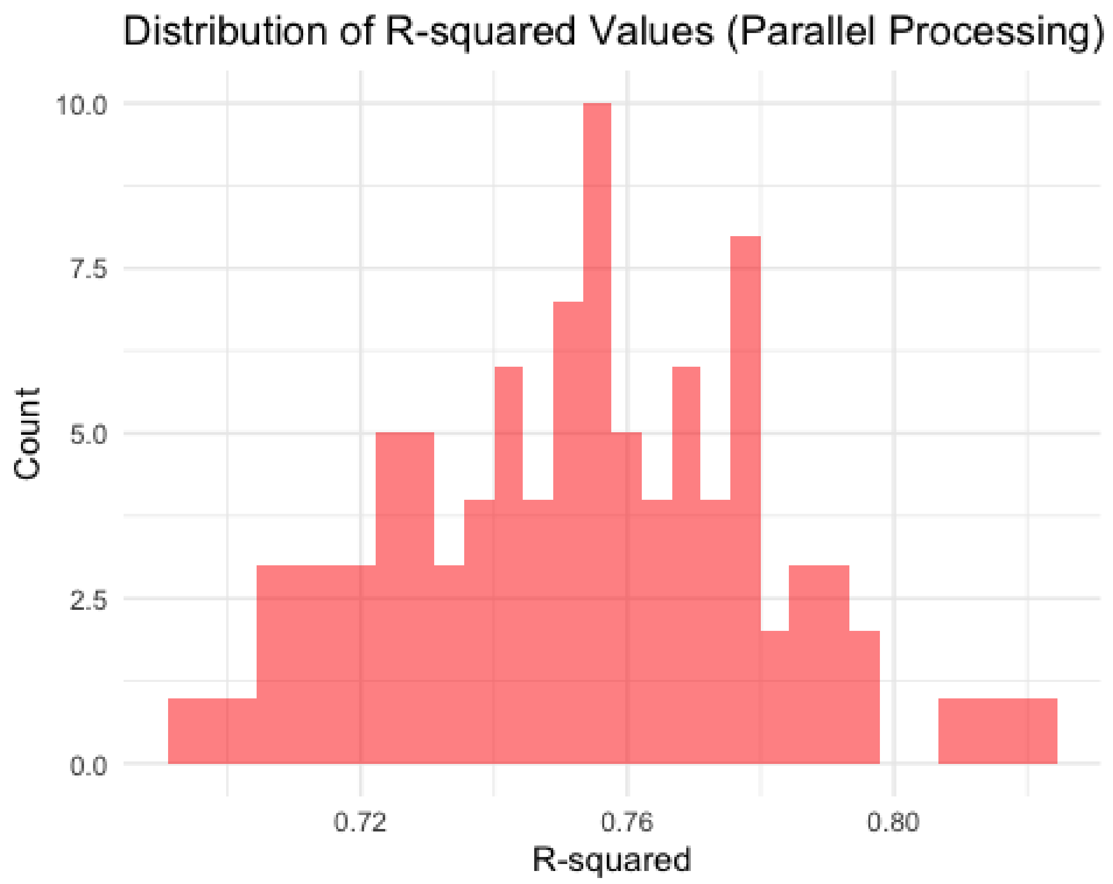
```r
glue("
Serial Processing Summary Statistics:
Mean R-squared: {serial_stats$mean}
Q1: {serial_stats$q1}
Q3: {serial_stats$q3}
IQR: {serial_stats$iqr}
")
```

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

Q4

```r
#Q4

# Set up parallel processing
#num_cores <- detectCores()  # Use all available cores
```

```r
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)
```

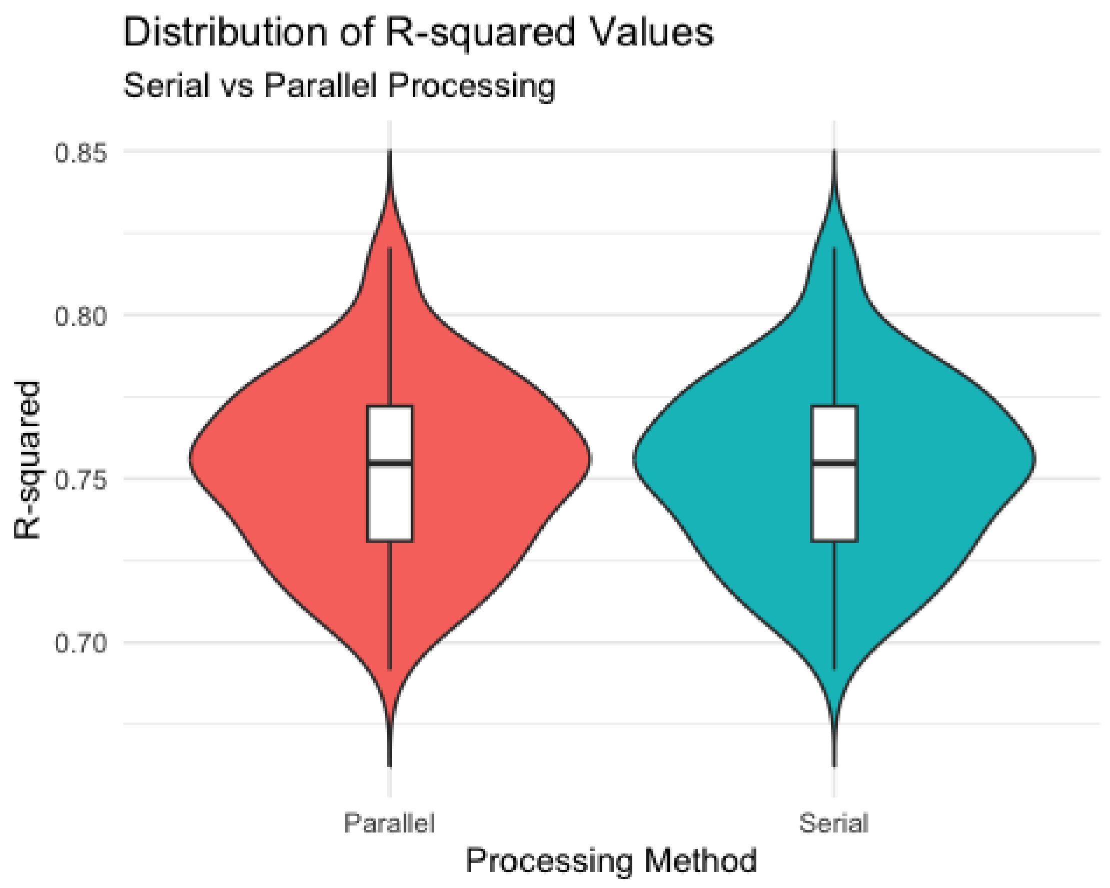


```
glue("
Parallel Processing Summary Statistics:
Mean R-squared: {parallel_stats$mean}
Q1: {parallel_stats$q1}
Q3: {parallel_stats$q3}
IQR: {parallel_stats$iqr}
")
```

```
## 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
```

```r
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.