Homework 5

GROUP 13

Problem 1

Introduction

Handling and analyzing large datasets is a common task in data science. However, processing a huge volume of data—such as generating and analyzing 100 million random numbers—can overwhelm system memory if not handled carefully. This report demonstrates a practical and scalable approach to processing large data in R by using parallel computing.

Objective

The aim is to:

- Efficiently simulate and analyze 100 million standard normal random numbers.
- Process data is chunks to avoid overflow.
- Use parallel computing to speed up execution.
- Compute a statistical summary(mean) of each chunk and overall dataset.

Methodology

To achieve our objective, the following steps are taken:

- **Chunking:** The entire dataset is divided into smaller manageable chunks(10 million values per chunk).
- **Parallelization:** Each chunk is processed using a separate core of the processor using the doParallel and foreach packages.
- **On-the-fly-Processing:** Each chunk is processed immediately after generation without saving all the numbers, reducing memory consumption.

- **Summary:** We compute the mean for each chunk and combine them to estimate the overall mean.

```r
library(parallel)
library(doParallel)
library(foreach)

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

chunk_size <- 1e7
total_numbers <- 1e8
total_chunks <- ceiling(total_numbers / chunk_size)

foreach(chunk = 1:total_chunks, .combine = c) %dopar% {
  start <- (chunk - 1) * chunk_size + 1
  end <- min(chunk * chunk_size, total_numbers)
  data <- rnorm(end - start + 1)
  mean(data)
} -> results

stopCluster(cl)
```

Conclusion

This report successfully demonstrates how to simulate and process **100 million** standard normal random values in R **without overwhelming system memory**. Key takeaways include:

- The use of **parallel computing** allowed the computation to finish faster by distributing the workload across multiple cores.
- **Chunk-wise processing** allowed memory-efficient handling of large data.
- The estimated overall mean was very close to the theoretical mean of the standard normal distribution , validating the approach.

Question 2: Empirical Estimation of Probability $||\mathbf{X}|| > 0.75$

Objective:

Estimate the empirical probability that the Euclidean norm of a random vector `X` exceeds 0.75. The vector `X` has a special covariance structure.

Objective:

Estimate the empirical probability that the Euclidean norm of a random vector `X` exceeds 0.75. The vector `X` has a special covariance structure.

Theoretical Background:

- Vector `X` ~ N(0, Σ), where Σ = K + 0.4I, and all off-diagonal entries of `K` are 0.6.
- Σ has eigenvalues:

- `= 0.4, multiplicity = (n-1)`

- `= 0.6n + 0.4, multiplicity = 1`

- A random vector with such a structure can be represented as `AX`, where `A` is orthogonal.
- Since orthogonal transformations preserve the norm ($\|AX\| = \|X\|$), we simulate the distribution accordingly.
- **Empirical Simulation:**
- **Approach:**

- `Repeat the experiment` `` `n_sim` `` `times (here, 10)`

- `For each simulation:`

  - `Generate (total_numbers - 1) i.i.d. samples from N(0, 0.4) in parallel.`

  - `Generate 1 sample from N(0, 0.6 \* total_numbers + 0.4).`

  - `Combine squared values and compute the norm.`

  - `Count if norm \> 0.75.`

- **Computation:**

- Parallel processing is again employed using `foreach` for efficiency.

- The squared norm is calculated from the sum of squares of generated values.

- Norm is computed and compared against the threshold (0.75).

- `ans` accumulates the number of times the norm exceeds 0.75.

- **Result:**

- The final probability is the average success count over `n_sim` runs.

```
library(parallel)

library(doParallel)
```

```
Warning: package 'doParallel' was built under R version 4.4.3
```

```
Loading required package: foreach
```

```
Warning: package 'foreach' was built under R version 4.4.3
```

```
Loading required package: iterators
```

```
Warning: package 'iterators' was built under R version 4.4.3
```

```
library(foreach)

# Setup parallel backend
num_cores <- detectCores() - 1  # Leave one core free
num_cores <-5
cl <- makeCluster(num_cores)
registerDoParallel(cl)

chunk_size <- 1e7
total_numbers <- 1e8
total_chunks <- ceiling(total_numbers / chunk_size)
```

```r
##We try to find the empirical probability of ||X||>0.75

#n_sim is the number of times we conduct the experiment
n_sim = 10
#ans will contain the number of time our proposition ||X||>0.75 is true
ans <- 0


for(sim in 1:n_sim) {

  ##Computing the sum of sq for 10^8 samples, in each process and storing it as a vector
  ##total_numbers-1 samples generated with mean = 0, var = 0.4

  chunk_norm <- foreach(chunk = 1:total_chunks, .combine = c) %dopar% {

    start <- (chunk - 1) * chunk_size + 1
    end <- min(chunk * chunk_size, total_numbers-1)

    numbers_in_chunk <- rnorm(end - start + 1 , 0 , sqrt(0.4))

    result <- sum(numbers_in_chunk^2)

    result
  }

  ##One sample with var = 0.6*n+0.4
  xn = rnorm(1 , 0 , sqrt(0.6*total_numbers+0.4))

  ##sum of sq of all Xi = square of the norm of vector X
  sq_norm <- sum(chunk_norm) + xn^2

  # Compute norm
  norm <- sqrt(sq_norm)

  # Return 1 if ||X|| > 0.75, 0 otherwise
  ans <- ans + as.numeric(norm > 0.75)

}

# Compute empirical probability

probability <- ans / n_sim
```

```
cat("Estimated P(||X|| > 0.75) =", probability, "\n")
```

```
Estimated P(||X|| > 0.75) = 1
```

Theoretical Probability Verification:

1. **Approach:**

- Use the weighted chi-squared distribution to theoretically compute P(‖X‖ \> 0.75)
- The squared norm follows a distribution of the form:\
  $\lambda_i \cdot \chi^2$, where λ_i are eigenvalues of Σ.

2. **Implementation:**

   - Use the `psum.chisq()` function from the `mgcv` package to compute the cumulative distribution function (CDF).
   - Set `lb` (eigenvalues) and `df` (degrees of freedom).

```
##Theoretically checking the probability
##Norm(X) follows weighted Chi-Squared(n) where n = number of samples
##Eigen values of the variance-covariance matrix are the weights

library(mgcv)
```

```
Loading required package: nlme
```

```
This is mgcv 1.9-1. For overview type 'help("mgcv-package")'.
```

```
# Define weights
lb <- c(0.6*total_numbers + 0.4, rep(0.4, total_numbers-1))

# Define degrees of freedom
df <- rep(1, total_numbers)

# Compute P(||X||²  0.75^2)
p_less_than <- psum.chisq(0.75^2, lb, df, lower.tail=TRUE)

# The answer is P(||X|| > 0.75) = 1 - p_less_than
```

```r
result <- 1 - p_less_than

cat("Theoretical probability of (||X|| > 0.75) is ", result , "\n")
```

```
Theoretical probability of (||X|| > 0.75) is  1
```

Conclusion

Both the **empirical** and **theoretical** probabilities of the event $||X|| > 0.75$ are equal to **1**, indicating that the norm of the random vector X almost surely exceeds 0.75 under the given distribution.

This outcome can be understood by analyzing the **eigenvalue structure** of the covariance matrix:

- The covariance matrix has one large eigenvalue, $\lambda_{\text{large}} = 0.6n + 0.4$, and $n - 1$ smaller eigenvalues, each equal to 0.4.
- Although the large eigenvalue contributes significantly to the squared norm of X, the cumulative contribution of the $n-1$ smaller eigenvalues is also substantial because their count grows with n.
- Specifically, the expected squared norm of X, being a weighted sum of chi-squared variables, scales approximately as:

$$\mathbb{E}[||X||^2] = (n-1) \cdot 0.4 + (0.6n + 0.4) = n \cdot 1 = n$$

- For $n = 10^8$, the expected norm $||X||$ is approximately $\sqrt{10^8} = 10^4$, which is **orders of magnitude larger than 0.75**.

Hence, the probability that $||X|| \leq 0.75$ is practically zero, making $P(||X|| > 0.75) = 1$. This result demonstrates how the **collective effect** of many small variances can dominate the behavior of high-dimensional random vectors, alongside any large individual variance.

The agreement between simulation and theory also confirms the accuracy of the modeling approach and the efficiency of parallel processing in handling high-dimensional probabilistic computations.