Tutorial: Parallel Simulations using R and Slurm

2025-11-13

Introduction

This tutorial demonstrates how to run a simulation study in parallel on a high-performance computing (HPC) cluster. Instead of running a loop 64 times sequentially on your laptop, we will submit 64 jobs simultaneously using Slurm Arrays.

Directory Structure

Before running these scripts, ensure your working directory contains the following folders to store outputs and logs:

- `/log` (For cluster standard output/error files)
- `/output` (For the simulation CSV results)

Step 1: Define Helper Functions (`01_function.R`)

First, we define the core logic of the simulation. This script contains functions to generate data and fit a linear model.

The Model: We generate data based on $y = 1 + \beta x + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$.

```r
# File: 01_function.R

regression_output <- function(data){
  model = lm(y ~ x, data = data)
  beta = as.numeric(coefficients(model)[2])
  beta_ci = confint(model, "x")
  sigma = summary(model)$sigma
  return(list(beta = beta, sigma = sigma, ci = beta_ci))
}

data_generate <- function(n = 100, beta = 1, sigma = 0.1){
  x <- rnorm(n, 0, 1)
  y <- 1 + beta*x + rnorm(n, 0, sigma)
  return(data.frame(x = x, y = y))
}
```

Note: This file does not run anything; it only defines tools used by the next script.

Step 2: The Driver Script (`02_sim_base.R`)

This is the main R script. It handles the parameter grid and detects which job is currently running.

The Parameter Grid

We create a grid of all possible combinations of sample size (n), effect size (β), and noise (σ).

```
params = expand.grid(n = c(64, 128, 256, 512),
                     beta = c(1, 2, 4, 8),
                     sigma = c(0.25, 0.5, 1, 2))
```

Since there are $4 \times 4 \times 4 = 64$ combinations, we will need 64 jobs.

The Handshake: Reading the Array ID

Crucially, this script accepts a command-line argument. This argument (an integer from 1 to 64) tells R which row of `params` to process.

```
## Interface with the cluster
args <- commandArgs(trailingOnly = TRUE)

if(length(args) == 0){
  i <- sample(1:nrow(params), 1) # Default for testing locally
} else {
  i <- as.numeric(args[1]) # Captures the value passed from Slurm
}

## Set parameters based on the job ID 'i'
n_true = params$n[i]
beta_true = params$beta[i]
sigma_true = params$sigma[i]
```

The script then runs the simulation `niter = 100` times for that specific parameter set and saves a uniquely named CSV file (e.g., `sim_1_...csv`) into the `output` folder.

Step 3: The Submission Script (`03_sim_shell.sh`)

This Bash script tells the Slurm scheduler how to run the jobs.

```
#!/bin/bash

#SBATCH --time=0-00:15
#SBATCH --mem-per-cpu=100MB
#SBATCH --array=1-64
#SBATCH --output=/path/to/log/slurm-%x_%A_%a.out

module load r

# Pass the array ID ($SLURM_ARRAY_TASK_ID) to R
Rscript /path/to/02_sim_base.R $SLURM_ARRAY_TASK_ID
```

Key Directives

- `#SBATCH --array=1-64`: This tells Slurm to launch this script 64 times.
- `$SLURM_ARRAY_TASK_ID`: This variable automatically changes (1, 2, ..., 64) for each instance of the job.
- The last line passes this ID into R, completing the connection between the cluster and your code.

Step 4: Aggregating Results (`04_summarize.R`)

Once all 64 jobs have finished, we use this script to combine the individual CSV files into one dataset.

```r
# File: 04_summarize.R

require(dplyr)
directory_name <- "/path/to/output"
files <- list.files(directory_name, full.names = TRUE)

## Read all simulation results into one dataframe
output <- files |> purrr::map_dfr(read.csv)

## Compute summary measures
results <- output |>
  group_by(beta_true, sigma_true, n) |>
  summarize(beta_hat = mean(beta_hat),
            sigma_hat = mean(sigma_hat),
            avg_runtime = mean(runtime),
            cov_prob = mean((beta_true - beta_lcb)*(beta_ucb - beta_true) > 0),
            .groups = "drop")
```

Workflow Summary

To execute the full simulation:

1. **Clean up:** Ensure the output folder is empty.
   ```
   rm output/*
   ```
2. **Submit:** Send the job array to the cluster.
   ```
   sbatch 03_sim_shell.sh
   ```
3. **Check Status:** Watch the queue.
   ```
   squeue -u <your_username>
   ```
4. **Analyze:** Once the queue is empty, run the summary script.
   ```
   Rscript 04_summarize.R
   ```