

Accelerating iteration in R with Parallel Processing

Sarita Singh

VeraMed



Abstract

As the clinical industry transitions toward wider adoption of R for statistical programming, optimizing run time of iterative operations becomes increasingly critical. Multiple imputation, bootstrapping, simulations serves as prime examples of the computational challenges posed by intensive, repetitive workflows in R.

Despite R's strengths in data analysis, it often struggles with runtime inefficiencies and memory limitations in large-scale, iterative tasks. In this work, we explore and benchmark various parallel processing strategies, including implementations using the parallel, future, and foreach frameworks, to significantly improve execution time. Using multiple imputation as a representative case study, we evaluate the trade-offs, scalability, and performance gains of each method. The findings provide practical guidance for selecting the most efficient approach based on dataset size and computing resources. These optimization strategies extend beyond imputation, offering broader applicability to other iterative functions in clinical data workflows and paving the way for more efficient R-based pipelines.

Why R Needs Parallel Computing

Because waiting for your code to finish shouldn't take longer than your clinical trial!

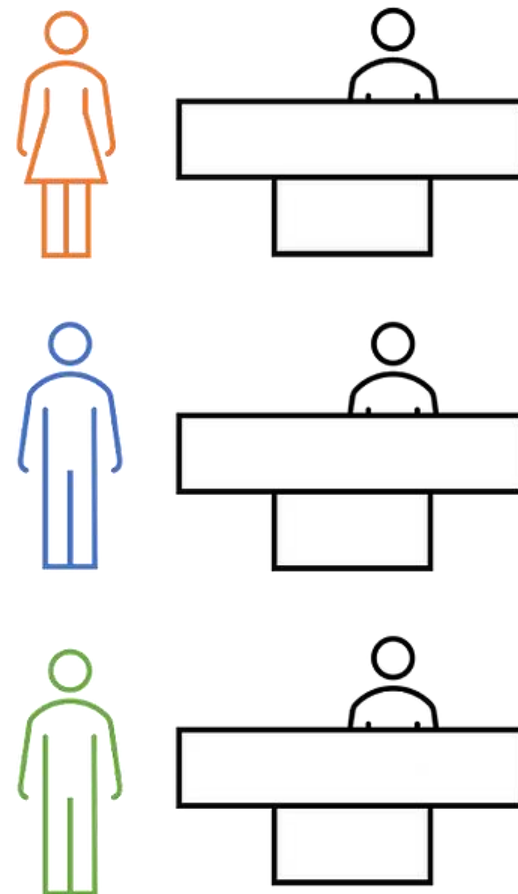
R is great at stats... but slow at laps when looping through big iterations

Multiple imputation and bootstrapping? More like multiple hours if you're running serially

Why run one job at a time when your CPU has more cores than a bag of apples?

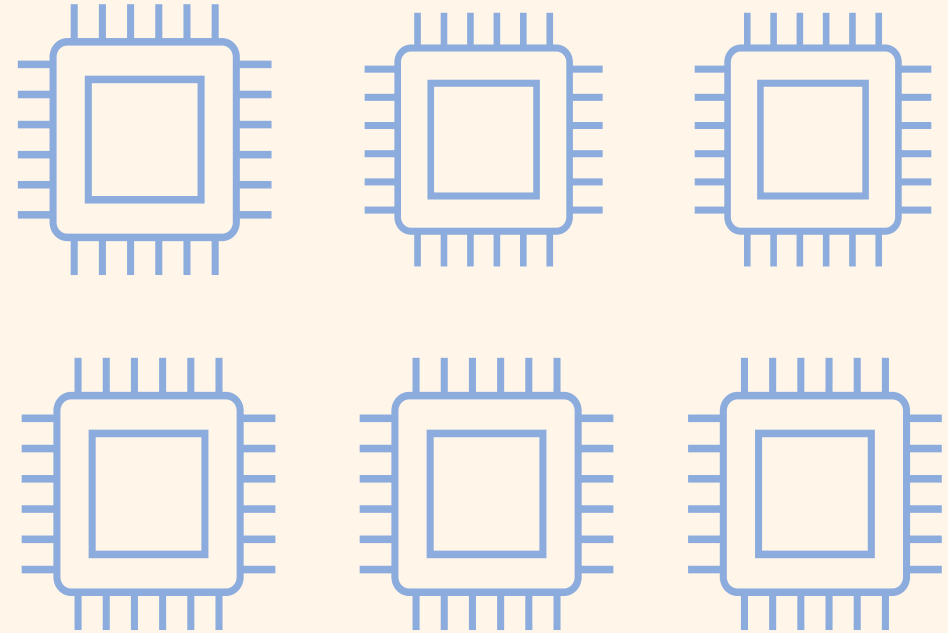
It's not just faster — it's R-responsible use of your resources!

Parallel Grocery Lines



Parallel Process

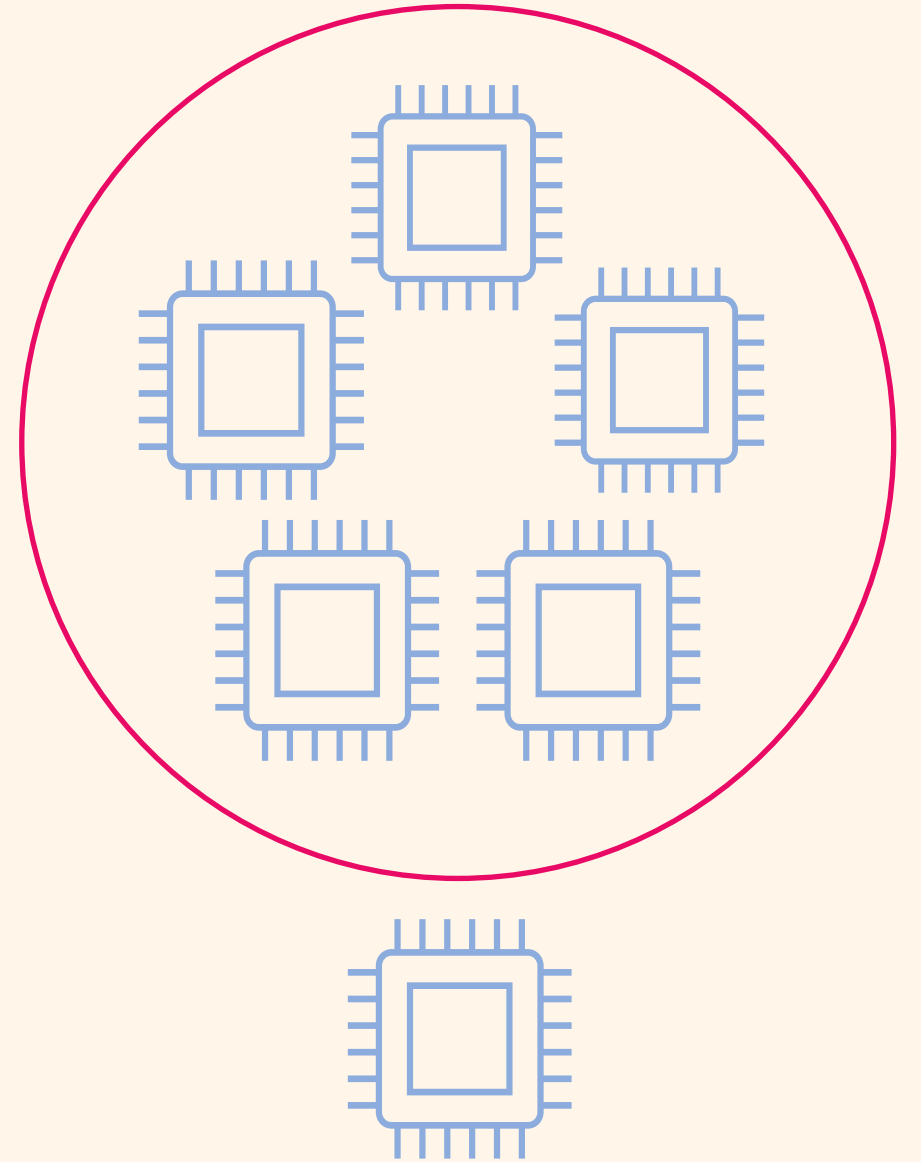
```
Library (parallel)  
detectCores()
```



Parallel Process

```
library (parallel)  
detectCores()
```

```
C1 <- makeCluster (detectCores() - 1)
```

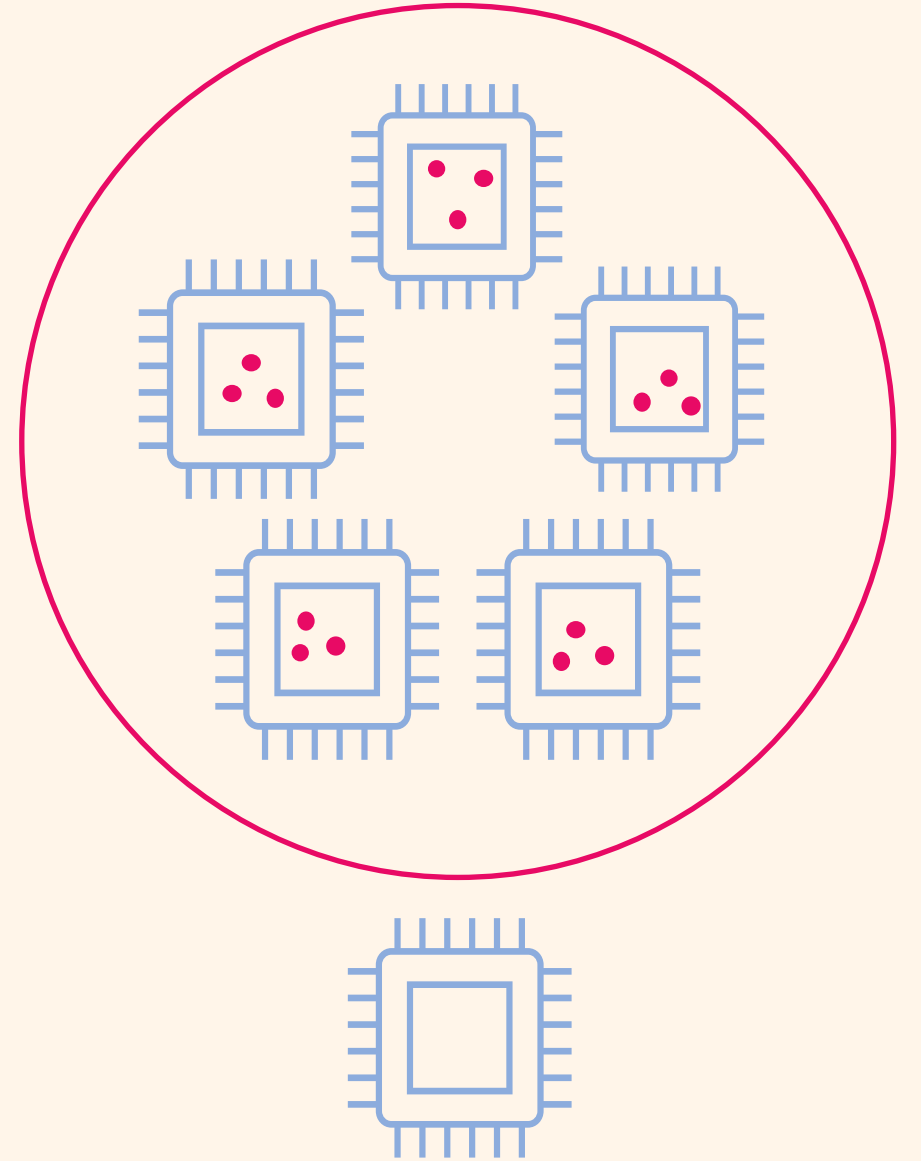


Parallel Process

```
library (parallel)  
detectCores()
```

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

```
clusterExport(cl, fun, .....)
```



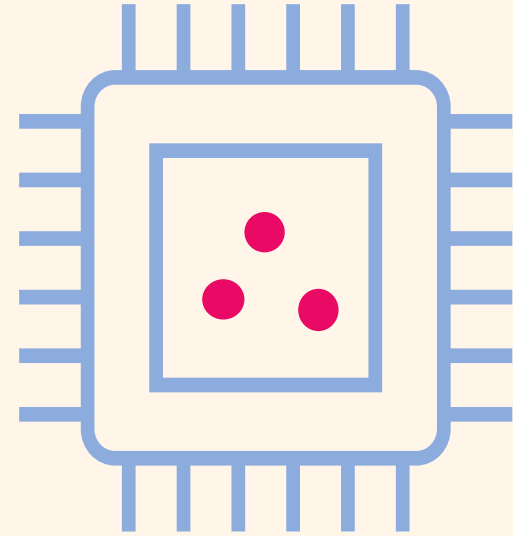
Parallel Process

```
library (parallel)  
detectCores()
```

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

```
clusterExport(cl, fun, .....)
```

```
stopCluster(cl)
```



We will



Perform Multiple imputation

- Sequential
- Parallel – Foreach + doParallel
- Parallel - future.apply

Benchmark performance using microbenchmark

Compare speed and structure

Case Data

NHANES example – all variables numerical

Description

A small data set with non-monotone missing values.

Format

- A data frame with 25 observations on the following 4 variables
- age - Age group (1=20-39, 2=40-59, 3=60+)
- bmi - Body mass index (kg/m**2)
- hyp - Hypertensive (1=no,2=yes)
- chl - Total serum cholesterol (mg/dL)

	age	bmi	hyp	chl
1	1	NA	NA	NA
2	2	22.7	1	187
3	1	NA	1	187
4	3	NA	NA	NA
5	1	20.4	1	113
6	3	NA	NA	184
7	1	22.5	1	118
8	1	30.1	1	187
9	2	22.0	1	238
10	2	NA	NA	NA
11	1	NA	NA	NA
12	2	NA	NA	NA
13	3	21.7	1	206
14	2	28.7	2	204
15	1	29.6	1	NA
16	1	NA	NA	NA
17	3	27.2	2	284
18	2	26.3	2	199
19	1	35.3	1	218
20	3	25.5	2	NA
21	1	NA	NA	NA
22	1	33.2	1	229
23	1	27.5	1	131
24	3	24.9	1	NA
25	2	27.4	1	186

Code Compare



```
# Load required libraries
library(mice)
library(microbenchmark)
library(foreach)
library(doParallel)
library(future.apply)
library(ggplot2)
library(tidyverse)

# ---- Example dataset ----
data <- nhanes
m <- 5

# Define seed list for reproducibility
set.seed(42)
seed_list <- 42 + 1:m
```

```
# ---- Parallel with foreach + doParallel ----
impute_foreach <- function(data, m, seed_list) {
  cl <- makeCluster(detectCores() - 1)
  registerDoParallel(cl)

  imp_list <- foreach(i = 1:m, .packages = "mice") %dopar% {
    set.seed(seed_list[i])
    mice(data, m = 1, maxit = 1000, printFlag = FALSE)
  }
  stopCluster(cl)
  registerDoSEQ()
  imp_combined <- Reduce(mice::ibind, imp_list)
  complete(imp_combined, action = "long", include = TRUE)
}
```

```
# ---- Sequential Imputation ----
impute_sequential <- function(data, m, seed_list) {
  imp_list <- list()
  for (i in 1:m) {
    set.seed(seed_list[i])
    imp_list[[i]] <- mice(data, m = 1, maxit = 1000, printFlag = FALSE)
  }
  imp_combined <- Reduce(mice::ibind, imp_list)
  complete(imp_combined, action = "long", include = TRUE)
}
```

```
# ---- Parallel with future.apply ----
impute_future <- function(data, m, seed_list) {
  plan(multisession, workers = availableCores() - 1)
  imps <- future_lapply(1:m, function(i) {
    set.seed(seed_list[i])
    mice(data, m = 1, maxit = 1000, printFlag = FALSE)
  }, future.seed = TRUE)
  imp_combined <- Reduce(mice::ibind, imps)
  complete(imp_combined, action = "long", include = TRUE)
}
```

Imputed Data



	age	bmi	hyp	chl	.imp	.id
26	1	26.3	1	187	1	1
27	2	22.7	1	187	1	2
28	1	22.0	1	187	1	3
29	3	22.5	1	186	1	4
30	1	20.4	1	113	1	5
31	3	21.7	1	184	1	6
32	1	22.5	1	118	1	7
33	1	30.1	1	187	1	8
34	2	22.0	1	238	1	9
35	2	22.7	1	187	1	10
36	1	29.6	1	229	1	11
37	2	22.0	1	229	1	12
38	3	21.7	1	206	1	13
39	2	28.7	2	204	1	14
40	1	29.6	1	187	1	15
41	1	29.6	1	199	1	16
42	3	27.2	2	284	1	17
43	2	26.3	2	199	1	18
44	1	35.3	1	218	1	19
45	3	25.5	2	206	1	20
46	1	30.1	1	184	1	21
47	1	33.2	1	229	1	22
48	1	27.5	1	131	1	23
49	3	24.9	1	218	1	24
50	2	27.4	1	186	1	25

	age	bmi	hyp	chl	.imp	.id
126	1	30.1	1	187	5	1
127	2	22.7	1	187	5	2
128	1	27.2	1	187	5	3
129	3	20.4	2	187	5	4
130	1	20.4	1	113	5	5
131	3	24.9	1	184	5	6
132	1	22.5	1	118	5	7
133	1	30.1	1	187	5	8
134	2	22.0	1	238	5	9
135	2	22.0	1	118	5	10
136	1	33.2	1	206	5	11
137	2	26.3	1	229	5	12
138	3	21.7	1	206	5	13
139	2	28.7	2	204	5	14
140	1	29.6	1	199	5	15
141	1	29.6	2	206	5	16
142	3	27.2	2	284	5	17
143	2	26.3	2	199	5	18
144	1	35.3	1	218	5	19
145	3	25.5	2	229	5	20
146	1	30.1	1	187	5	21
147	1	33.2	1	229	5	22
148	1	27.5	1	131	5	23
149	3	24.9	1	229	5	24
150	2	27.4	1	186	5	25

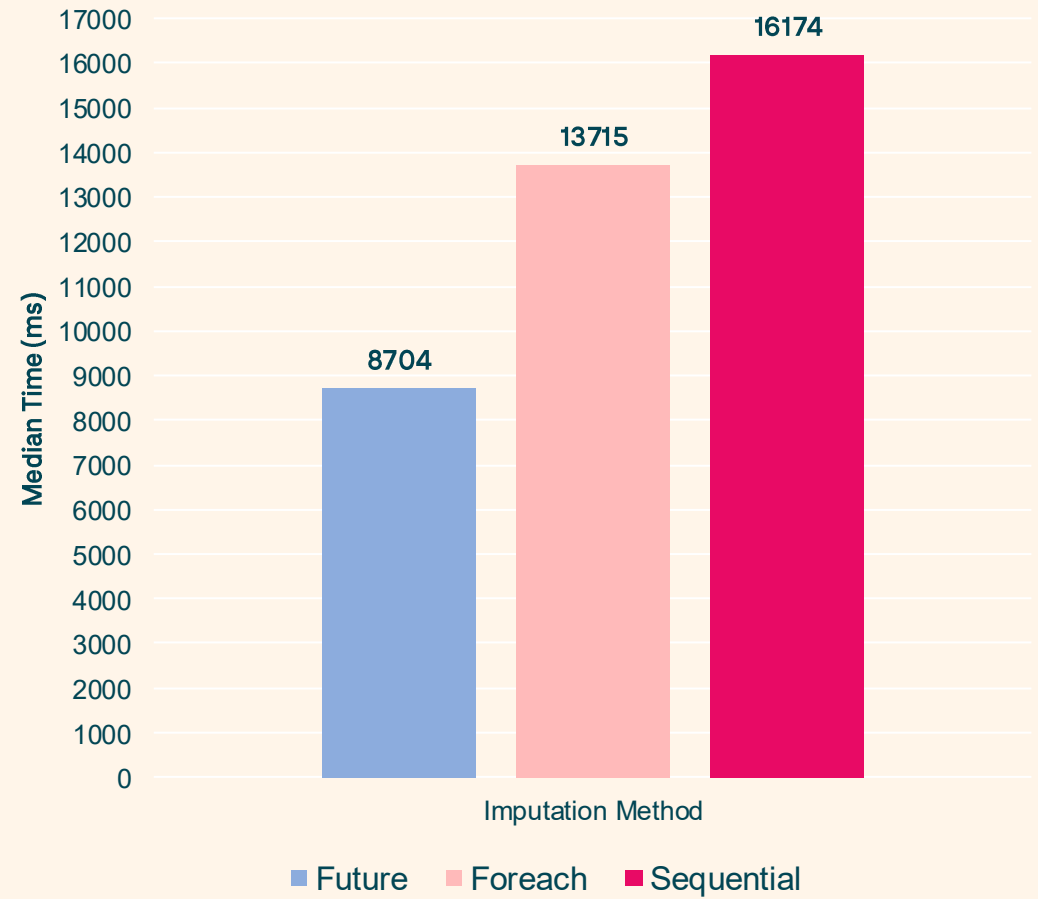
Benchmarking

```
# ---- Benchmarking ----  
benchmark_results <- microbenchmark(  
  sequential = {imp_seq <- impute_sequential(data, m, seed_list)},  
  foreach = {imp_foreach <- impute_foreach(data, m, seed_list)},  
  future = {imp_future <- impute_future(data, m, seed_list)},  
  times = 3L  
)
```

	expr	time
1	sequential	16173881201
2	foreach	19615221300
3	future	6013282900
4	sequential	15173286201
5	foreach	13715002201
6	sequential	19524052902
7	foreach	12940801301
8	future	9102504801
9	future	8704444300

```
# ---- Summary Table ----  
summary_df <- data.frame(  
  Method = c("Sequential", "foreach", "future"),  
  Median_Time_ms = tapply(benchmark_results$time, benchmark_results$expr, median) / 1e6  
)
```

Benchmarking Imputation Methods

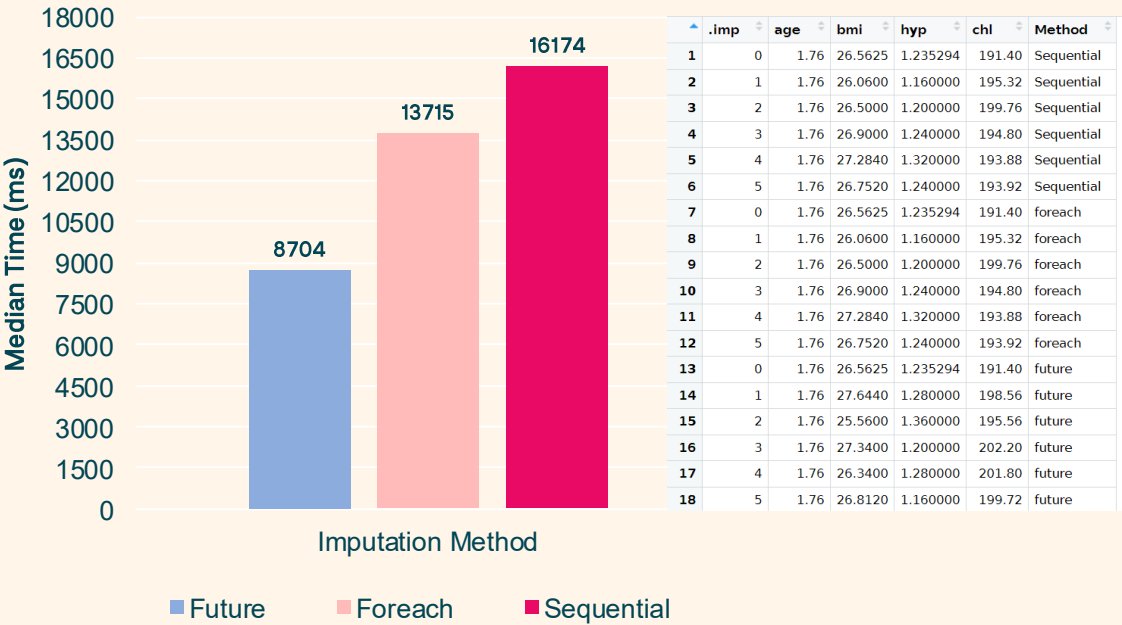


Imputed Data Summary

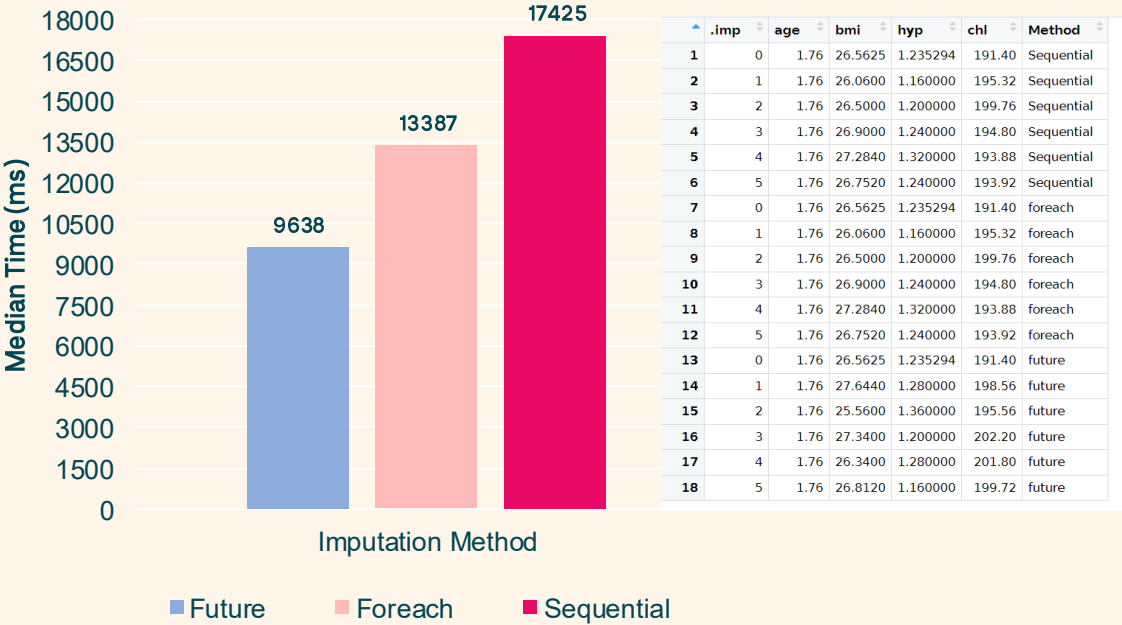
Benchmarking Imputation Methods



Iteration 1



Iteration 10



Debugging in Parallel environment



Mimic Sequential Execution: makeCluster(1) →
use normal debugging tools

```
library(parallel)
cl <- makeCluster(1)
# run your clusterApply/ parLapply work
# here
stopCluster(cl)

# With foreach/doParallel
library(doParallel)
cl <- makeCluster(1)
registerDoParallel(cl)
# foreach(...) %dopar% { ... }
stopCluster(cl)
registerDoSEQ()
```

Log Outputs: outfile="log.txt" or use
ParallelLogger

```
library(parallel)
cl <- makeCluster(4, outfile =
"cluster.log")
# All print()/message()/warnings from
# workers go to cluster.log
# ... work ...
stopCluster(cl)
```

Debugging in Parallel environment

Inspect Variables

Add print()/message(); reduce iterations

```
# Example within a foreach loop
foreach(i = 1:M, .packages =
c("mice")) %dopar% {
  message(sprintf("Starting
iteration %d", i))
  # ... work ...
  message(sprintf("Completed
iteration %d", i))
}
```

Interactive Debugging

partools::dbs() for browser() in workers

```
library(partools)
dbs() # start debuggable
cluster sessions
# Inside worker code, call
browser() at points of
interest
```

Reproducibility

Set seeds
(future.seed=TRUE)

Error Handling

stopOnError=TRUE or
.errorhandling="pass"

```
# Stop on first error
(parallel::clusterApply)
clusterApply(cl, X, FUN,
stopOnError = TRUE)
```



Discussion/Summary

no single method is universally optimal — best choice depends on context

Benchmark to validate performance

- Iterative operations like multiple imputation can be significantly accelerated
 - Parallel frameworks (e.g., parallel, future, foreach) reduce runtime, especially with large datasets or many iterations
 - Parallelization leverages multi-core systems, improving throughput without increasing complexity
 - Multiple frameworks support parallelization with varying ease of integration and overhead
 - Strategies tested for imputation are adaptable to other repetitive workflows (e.g., bootstrapping, simulations)
-

Caution Areas – Avoid parallel overhead for small tasks

References

R Programming for Data Science: Roger D Peng - Ch 18, 22

<https://bookdown.org/rdpeng/rprogdatascience/>

<https://bookdown.org/rdpeng/rprogdatascience/debugging.html>

Multiple Imputation

<https://dept.stat.lsa.umich.edu/~jerrick/courses/stat701/notes/mi.html>

Advanced R by Hadley Wickham

<http://adv-r.had.co.nz/Profiling.html>

Stackoverflow

<https://stackoverflow.com/questions/24040280/parallel-computation-of-multiple-imputations-by-using-mice>

Flexible Imputation of missing data

<https://stefvanbuuren.name/fimd/>

Berkeley Statistics

<https://computing.stat.berkeley.edu/tutorial-parallelization/parallel-R.html>

Posit

<https://support.posit.co/hc/en-us/articles/205612627-Debugging-with-the-RStudio-IDE>

R-Bloggers

<https://www.r-bloggers.com/2017/08/implementing-parallel-processing-in-r>

mice R documentation

<https://www.rdocumentation.org/packages/mice/versions/3.19.0>

microbenchmark R documentation

<https://www.rdocumentation.org/packages/microbenchmark/versions/1.5.0>

Foreach R documentation:

<https://www.rdocumentation.org/packages/foreach/versions/1.5.2>

doParallel R documentation :

<https://www.rdocumentation.org/packages/doParallel/versions/1.0.17>

future.apply R documentattion:

<https://www.rdocumentation.org/packages/future.apply/versions/1.11.2/topics/future.apply>

makeCluster R documentation

<https://www.rdocumentation.org/packages/parallel/versions/3.6.2/topics/makeCluster>

ParallelLogger R documentation:

<https://www.rdocumentation.org/packages/ParallelLogger/versions/3.5.1>

partools R documentation:

<https://www.rdocumentation.org/packages/partools/versions/1.1.6>



Thank you