Homework 2: A simulation study investigating the bootstrap

BIOS 731 – Advanced Statistical Computing

Context and learning objectives

This assignment reinforces ideas in Module 2: **Simulations and Resampling Methods**. We focus specifically on implementing a **large-scale simulation study**, and the assignment also includes components involving the bootstrap, parallelization, Git/GitHub, and project organization.

Due Date and Submission

Submit (via Canvas) a PDF knitted from a `.Rmd` or `.qmd` file. Your PDF should include the web address of the GitHub repository containing your work for this assignment. **Commits after the due date will cause the assignment to be considered late.**

Point distribution

Problem	Points
Problem 0	20
Problem 1.1	10
Problem 1.2	5
Problem 1.3	20
Problem 1.4	30
Problem 1.5	15

Problem 0

This "problem" focuses on the structure of your submission, especially the **use of git and GitHub** for reproducibility, **R Projects** to organize your work, **Quarto/R Markdown** to write reproducible reports, **relative paths** to load local files, and reasonable naming conventions for your files.

To that end:

- Create a public GitHub repository and a local R Project. I suggest naming the repo/directory `bios731_hw2_YourLastName` (e.g., `bios731_hw2_wrobel`).
- Push your **entire project folder** to GitHub.
- Submit a PDF knitted from your `.Rmd`/`.qmd` file to Canvas.
  - Your solutions should be implemented in your `.Rmd`/`.qmd` file.
  - Your git commit history should reflect your workflow (i.e., multiple meaningful commits; avoid a single "final" commit with all work).

Problem 1

Simulation study. Your goal in this homework is to plan and execute a well-organized simulation study for multiple linear regression and confidence intervals constructed via both Wald and bootstrap methods.

Model

Consider the multiple linear regression model

$$Y_i = \beta_0 + \beta_{treatment} X_{i1} + \mathbf{Z_i}^T \gamma + \epsilon_i$$

where we are primarily interested in the treatment effect $\beta_{treatment}$. It is fine to simulate data with no confounders, i.e. $\gamma = 0$.

Notation

Notation is defined below:

- Y_i: continuous outcome
- X_{i1}: treatment group indicator; $X_{i1} = 1$ for treated
- $\mathbf{Z_i}$: vector of potential confounders
- $\beta_{treatment}$: average treatment effect, adjusting for $\mathbf{Z_i}$
- γ: vector of regression coefficient values for confounders

- ϵ_i: errors, we will vary how these are defined

Simulation goals

In our simulation, we want to

- Estimate $\beta_{treatment}$ and $se(\hat{\beta}_{treatment})$
  - Evaluate $\beta_{treatment}$ through bias and coverage

You will compare **three** methods for constructing a 95% confidence interval for $\hat{\beta}_{treatment}$:

1. Wald confidence intervals (standard model-based approach)
2. Nonparametric bootstrap **percentile** intervals
3. Nonparametric bootstrap t intervals

You will also evaluate **computation time** for each method.

Simulation design (full factorial)

Evaluate performance across the following factors:

- Sample size: $n \in \{10, 50, 500\}$
- True treatment effect: $\beta_{treatment} \in \{0, 0.5, 2\}$
- Error distribution:
  - Normal errors: $\epsilon_i \sim N(0, 2)$
  - Heavy-tailed errors: $\epsilon_i \sim t_\nu$ with $\nu = 3$, scaled to have variance 2

Implementation hint (heavy tails). If $u \sim t_{nu}$ with $\nu > 2$, then $\text{Var}(u) = \nu/(\nu - 2)$. To match the normal-error condition variance, set

$$\epsilon_i = u \cdot \sqrt{2\,\frac{\nu - 2}{\nu}}.$$

Problem 1.1 ADEMP Structure

Answer the following questions. Use the ADEMP framework explicitly:

- **A (Aim):** What is the goal of the simulation study?
- **D (Data-generating mechanism):** What model and distributions generate the data? What factors vary across scenarios?
- **E (Estimand):** What quantity(ies) are you trying to learn about?
- **M (Methods):** What methods are being evaluated/compared?
- **P (Performance measures):** What metrics summarize performance?

Also answer:

- How many simulation scenarios will you be running (i.e., how many unique combinations in the full factorial design)?

My answers:

- **A (Aim):** The aim of this simulation study is to evaluate the finite-sample performance of confidence interval procedures for the treatment effect parameter $\beta_{\text{treatment}}$ in a linear regression model.

Specifically, we assess and compare the bias, coverage probability, and variability of standard error estimates for Wald confidence intervals and two nonparametric bootstrap-based confidence intervals (percentile and bootstrap-t) under varying sample sizes, true treatment effects, and error distributions.

- **D (Data-generating mechanism):** We generate data according to the linear regression model

$$Y_i = \beta_0 + \beta_{\text{treatment}} X_{i1} + \epsilon_i$$

  where X_{i1} is a binary treatment indicator, and ϵ_i are error terms. For simplicity, no additional confounders are included in the data-generating mechanism. We vary the following factors across simulation scenarios:

- Sample size: $n \in \{10, 50, 500\}$
- True treatment effect: $\beta_{\text{treatment}} \in \{0, 0.5, 2\}$
- Error distribution:
  - Normal errors: $\epsilon_i \sim N(0, 2)$
  - Heavy-tailed errors: $\epsilon_i \sim t_\nu$ with $\nu = 3$, scaled to have variance 2

- **E (Estimand):** The primary estimand of interest is the treatment effect parameter $\beta_{\text{treatment}}$ in the linear regression model. We also focus on the standard error of the estimator $\widehat{\beta}_{\text{treatment}}$.
- **M (Methods):** We evaluate three methods for constructing 95% confidence intervals for $\widehat{\beta}_{\text{treatment}}$:

1. Wald confidence intervals based on the standard error from the fitted linear model.
2. Nonparametric bootstrap percentile intervals, which use the empirical distribution of $\widehat{\beta}^*_{\text{treatment}}$ from bootstrap resamples.
3. Nonparametric bootstrap t intervals, which standardize the bootstrap estimates using their estimated standard errors from a second level of bootstrapping.

- **P (Performance measures):** We summarize performance using the following metrics:

1. Bias of $\widehat{\beta}_{\text{treatment}}$: the average difference between the estimated and true treatment effect across simulations.
2. Coverage probability of the 95% confidence intervals: the proportion of simulations in which the interval contains the true treatment effect.
3. Distribution of the estimated standard error $se(\widehat{\beta}_{\text{treatment}})$ across simulations.
4. Computation time for each method, measured in seconds.

- **Number of simulation scenarios:** There are 3 levels of sample size, 3 levels of true treatment effect, and 2 levels of error distribution, resulting in a total of $3 \times 3 \times 2 = 18$ unique simulation scenarios.

Problem 1.2 nSim

Based on desired coverage of 95% with Monte Carlo error of no more than 1%, how many simulations (n_{sim}) should you perform for **each** simulation scenario? Implement this value of n_{sim} throughout your simulation study.

My answer:

To achieve a Monte Carlo error of no more than 1% for estimating the coverage probability of 95%, we can use the formula for the standard error of a proportion:

$$SE = \sqrt{\frac{p(1-p)}{n_{sim}}},$$

where p is the true coverage probability (0.95 in this case). We want this standard error to be less than or equal to 0.01:

$$\sqrt{0.95(1-0.95)/n_{sim}} \leq 0.01,$$

which yields $n_{sim} \geq 475$. To be conservative, we can round up to $n_{sim} = 500$ simulations per scenario to ensure that the Monte Carlo error is within the desired threshold.

Problem 1.3 Implementation

For bootstrap t, goal is to estimate a t distribution given by

$$t^* = \frac{\hat{\theta}^* - \hat{\theta}}{s_{\hat{\theta}^*}}$$

where

- $\hat{\theta}^*$: estimated parameter value from each bootstrap iteration
- $\hat{\theta}$: parameter estimate from the original sample
- $s_{\hat{\theta}^*}$ standard error estimate from a given bootstrap sample; requires a second level of bootstrapping to construct.

My implementation:

To improve computational efficiency, simulation scenarios were parallelized on a computing cluster using Slurm array jobs. For each scenario (defined by sample size, true treatment effect, and error distribution), a shell script (submit_array.sh) was used to submit an array job, where each task ran a single replicate by calling an R script (run_rep.R). Each replicate saved its results as an intermediate .rds file in a scenario-specific subfolder under data/.

After all simulation scenarios completed, the intermediate results were downloaded to a local machine and aggregated in a post-processing step using a separate R script. This script combined replicate-level outputs into scenario-level summaries, which were then used to generate all tables and figures reported in the analysis.

All file paths were specified using relative paths, and intermediate simulation outputs were excluded from version control via .gitignore to ensure a clean and reproducible project structure.

Parameter choices

- Use $\mathbf{B = 500}$ outer bootstrap resamples for the percentile and bootstrap-t intervals.
- For bootstrap-t, use $\mathbf{B\_inner = 100}$ inner bootstrap resamples.
- Construct **95%** confidence intervals for all methods.

(If you choose different values, justify your choice and discuss the computation/accuracy trade-off.)

Computing + reproducibility requirements

Execute the full simulation study. For full credit, implement the following:

- Well-structured scripts and subfolders following guidance from the `project_organization` lecture
- Use relative file paths to access intermediate scripts and data objects
- Use readable code practices (clear function boundaries, meaningful names, minimal duplication)
- **Parallelize across simulation scenarios**
- Save results from each simulation scenario to an intermediate `.Rds` or `.Rda` file in a `data/` subfolder
  - Add these files to `.gitignore` so they are not pushed to GitHub
- Include a `README.md` explaining your workflow
  - Include what files to run, in what order, and how outputs are produced
- Ensure end-to-end reproducibility:
  - I should be able to clone your GitHub repo, open your `.Rproj`, and run the simulation study to regenerate results

Problem 1.4 Results summary

Create a plot or table summarizing simulation results across scenarios and methods for each of the following:

- Bias of $\hat{\beta}$
- Coverage of the **95% CI** for $\hat{\beta}$
- Distribution of $se(\hat{\beta})$
- Computation time across methods

Presentation guidance

- If creating plots, I encourage faceting by at least one design factor (e.g., n, error distribution, or true treatment value).
- Include informative captions for each plot/table.
- For coverage plots, consider adding a reference line at 0.95.

```
# A tibble: 18 x 17
       n  beta err    n_rep      bias emp_se cov_wald cov_pct cov_bt len_wald
   <int> <dbl> <chr>  <int>     <dbl>  <dbl>    <dbl>   <dbl>  <dbl>    <dbl>
 1    10   0   normal   495  0.00704   0.953    0.913   0.860  0.982    3.64
 2    10   0   t3       500  0.0230    1.03     0.932   0.87   0.988    3.36
 3    10   0.5 normal   500  0.0381    0.957    0.912   0.866  0.982    3.62
 4    10   0.5 t3       500  0.000460  0.937    0.94    0.892  0.98     3.24
 5    10   2   normal   500 -0.00729   0.940    0.918   0.862  0.988    3.63
 6    10   2   t3       500 -0.00721   1.05     0.89    0.824  0.97     3.12
 7    50   0   normal   500  0.0211    0.420    0.942   0.934  0.948    1.57
 8    50   0   t3       500 -0.0221    0.416    0.944   0.9    0.91     1.52
 9    50   0.5 normal   500 -0.00315   0.424    0.924   0.924  0.938    1.57
10    50   0.5 t3       500 -0.00668   0.396    0.946   0.932  0.93     1.50
11    50   2   normal   500 -0.00594   0.407    0.938   0.93   0.938    1.57
12    50   2   t3       500 -0.0151    0.408    0.942   0.93   0.936    1.49
13   500   0   normal   500 -0.00842   0.123    0.964   0.96   0.962    0.497
14   500   0   t3       500  0.00393   0.122    0.944   0.942  0.942    0.482
15   500   0.5 normal   500 -0.00311   0.130    0.948   0.944  0.948    0.496
16   500   0.5 t3       500 -0.00964   0.123    0.95    0.944  0.944    0.481
17   500   2   normal   500  0.00325   0.130    0.946   0.95   0.946    0.496
18   500   2   t3       500  0.00329   0.121    0.958   0.958  0.954    0.488
# i 7 more variables: len_pct <dbl>, len_bt <dbl>, time_wald <dbl>,
#   time_pct <dbl>, time_bt <dbl>, pct_fail <dbl>, bt_fail <dbl>
```

```r
### bias of hat beta
ggplot(summary, aes(x = n, y = bias, group = 1)) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  geom_point() +
  geom_line() +
  facet_grid(err ~ beta) +
  labs(
    title = "Bias of hat beta across simulation scenarios",
    x = "Sample size (n)",
    y = "Bias: E[ - ]"
  ) +
  theme_bw()
```

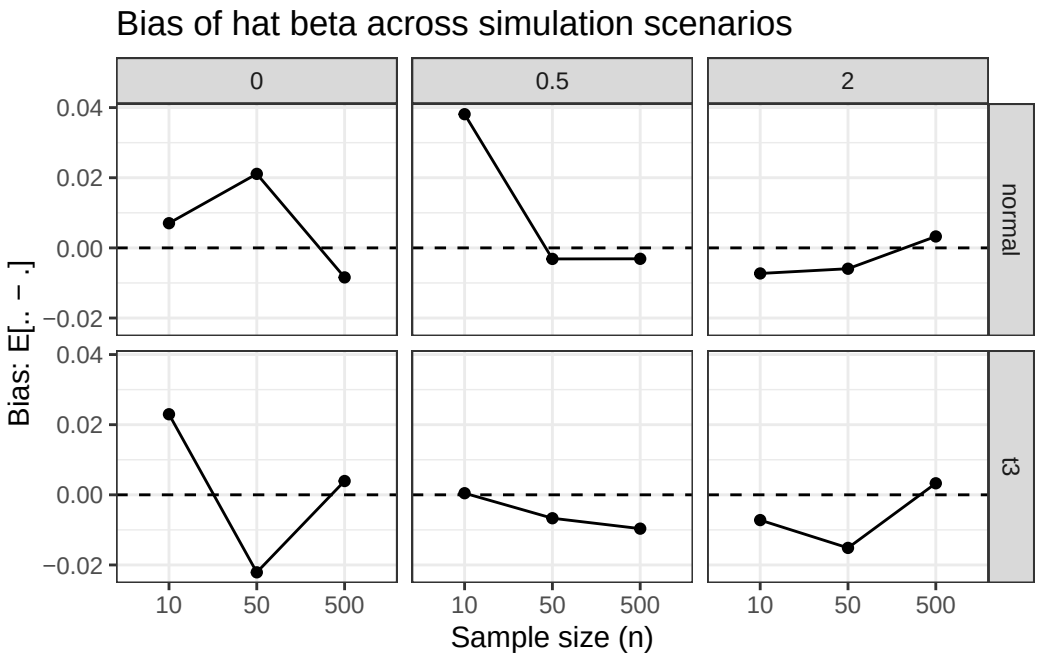

Figure 1: Bias of the treatment effect estimator hat beta_treatment$ across simulation scenarios. The dashed line indicates zero bias.

```r
# read replicate-level se_hat from all scenarios
rep_dirs <- list.dirs("data", recursive = FALSE)
read_se_one_dir <- function(dir) {
  files <- list.files(dir, full.names = TRUE)
  sims <- do.call(rbind, lapply(files, readRDS))

  nm <- basename(dir)
  parts <- strsplit(nm, "_")[[1]]
  n <- as.integer(sub("n", "", parts[1]))
  beta <- as.numeric(sub("beta", "", parts[2]))
  err <- parts[3]

  sims %>%
    transmute(
      n = factor(n, levels = c(10, 50, 500)),
      beta = factor(beta, levels = c(0, 0.5, 2)),
      err = factor(err, levels = c("normal", "t3")),
      se_hat = se_hat
    )
```

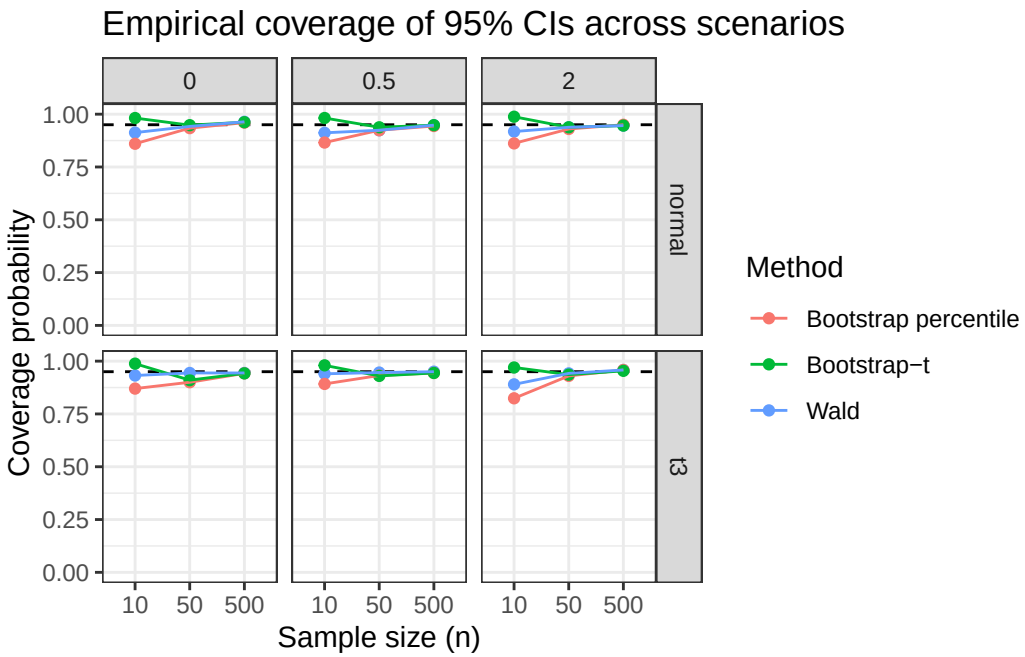

Figure 2: Empirical coverage of nominal 95% confidence intervals across simulation scenarios and methods. The dashed line indicates the nominal coverage level of 0.95.

```
}

se_df <- do.call(rbind, lapply(rep_dirs, read_se_one_dir)) %>%
  filter(is.finite(se_hat))

ggplot(se_df, aes(x = se_hat)) +
  geom_histogram(bins = 30) +
  facet_grid(err ~ beta) +
  labs(
    title = "Distribution of se(^) across replicates",
    x = "se(^) from lm()",
    y = "Count"
  ) +
  theme_bw()
```

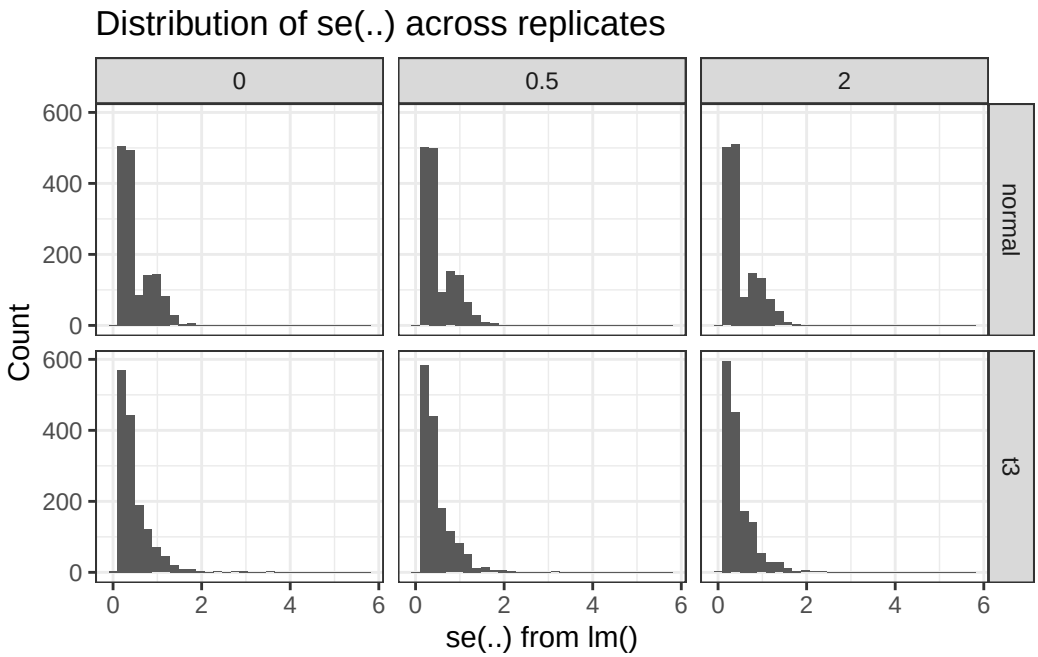

Figure 3: Distribution of the model-based standard error estimates se(ˆ) across Monte Carlo replicates. Faceted by error distribution and true treatment effect; shown separately for each sample size.

```
time_long <- summary %>%
  select(n, beta, err, time_wald, time_pct, time_bt) %>%
  pivot_longer(
    cols = starts_with("time_"),
    names_to = "method",
    values_to = "time_sec"
  ) %>%
  mutate(
    method = recode(method,
      time_wald = "Wald",
      time_pct  = "Bootstrap percentile",
      time_bt   = "Bootstrap-t"
    )
  )

ggplot(time_long, aes(x = n, y = time_sec, color = method, group = method)) +
  geom_point() +
  geom_line() +
  facet_grid(err ~ beta) +
```

```
  scale_y_log10() +
  labs(
    title = "Computation time per replicate across CI methods",
    x = "Sample size (n)",
    y = "Mean time (seconds, log scale)",
    color = "Method"
  ) +
  theme_bw()
```

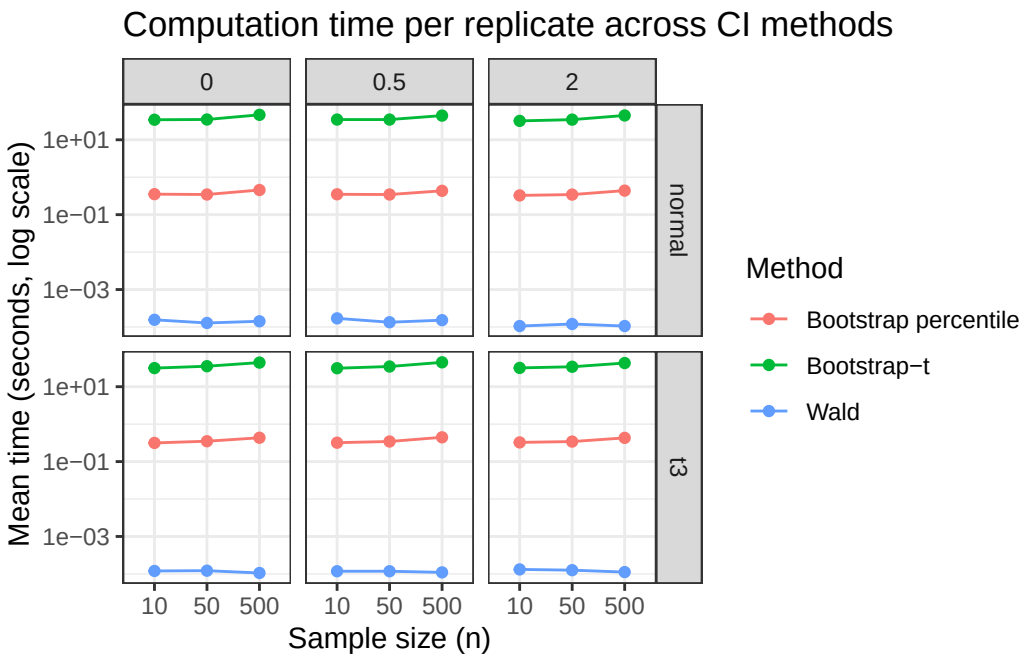

Figure 4: Average computation time per replicate for each CI method across scenarios. Times are measured within each replicate using system.time and then averaged across completed replicates. Y-axis uses log scale for readability.

Problem 1.5 Discussion

Interpret the results summarized in Problem 1.4.

1. Write **one paragraph** summarizing the main findings of your simulation study.
2. Then answer the questions below:

- How do the different methods for constructing confidence intervals compare in terms of computation time?

- Which method(s) provide the best coverage when $\epsilon_i \sim N(0, 2)$?
- Which method(s) provide the best coverage for the heavy-tailed errors?

Finally, briefly comment on any notable interactions (e.g., how performance changes with n or error type) and any practical recommendations you would make based on your results.

My discussion:

1. Overall, the simulation study suggests that the OLS estimator of the treatment effect is essentially unbiased across all scenarios, with any finite-sample bias being small and shrinking as the sample size increases. Empirical coverage of nominal 95% confidence intervals improves with larger n and is generally close to 0.95 for $n = 500$ under both error distributions. In small samples ($n = 10$), coverage is noticeably method-dependent: the bootstrap-t interval tends to have the best (often slightly conservative) coverage, while the percentile bootstrap is most prone to under-coverage; the Wald interval typically falls in between and improves quickly as n increases. The estimated standard errors $se(\widehat{\beta})$ show a right-skewed distribution, with heavier tails leading to greater variability, especially at small n. Computation time differs dramatically across methods: Wald intervals are essentially instantaneous, the percentile bootstrap is substantially slower, and bootstrap-t is by far the most computationally intensive.

2.1 computation time

The Wald interval is the fastest method by orders of magnitude, since it only requires one model fit and an analytic standard error. The percentile bootstrap is slower because it refits the model many times (one bootstrap loop), but remains far more feasible than bootstrap-t. Bootstrap-t is the slowest by a large margin because it requires an outer bootstrap plus an inner bootstrap to estimate the standard error within each outer sample (a nested bootstrap), which leads to much larger computational cost. This pattern is consistent across all scenarios and does not materially depend on the true treatment effect; any changes with n are small compared with the method-driven differences.

2.2 best coverage under normal errors

Under normal errors, all three methods approach the nominal 0.95 coverage as n increases, and by $n = 500$ the differences are minimal. In the smallest sample size, the bootstrap-t interval generally provides the best coverage (often slightly above 0.95, indicating mild conservatism), while the Wald interval is reasonably close and improves quickly with n. The percentile bootstrap shows the most under-coverage at $n = 10$, but it also converges toward 0.95 as n increases.

2.3 best coverage under heavy-tailed errors

With heavy-tailed errors, the advantage of bootstrap-t is more apparent in small samples: it tends to maintain coverage closest to (or slightly above) 0.95, reflecting additional robustness

from studentizing. The Wald interval also improves with n and is close to nominal for moderate to large sample sizes, but is less reliable at $n = 10$. The percentile bootstrap again shows the most pronounced under-coverage in the smallest sample size, suggesting that the percentile method is more sensitive to skewness/heavy tails in finite samples.

2.4 interactions and practical recommendations

The most notable interaction is between sample size and error distribution: heavy tails increase the variability of $se(\widehat{\beta})$ and make small-sample inference harder, which is where the choice of CI method matters most. Method differences largely vanish as n grows, consistent with asymptotic normality of $\widehat{\beta}$. Practically, if computation is a major constraint (or n is moderate/large), the Wald interval is a strong default because it is extremely fast and performs well once n is not tiny. If the sample size is very small and/or the error distribution may be heavy-tailed, bootstrap-t provides the most reliable coverage but at substantial computational cost, making it best used when accuracy is prioritized over runtime. The percentile bootstrap is attractive for its simplicity, but based on these results it is the least reliable in small samples (especially under heavy tails) and should be used cautiously when accurate coverage is important.