Homework 1

Context

This assignment reinforces ideas in Module 1: Reproducible computing in R. We focus specifically on implementing a large scale simulation study, but the assignment will also include components involving bootstrap and parallelization, Git/GitHub, and project organization.

Due date and submission

Please submit (via Canvas) a PDF knitted from .Rmd. Your PDF should include the web address of the GitHub repo containing your work for this assignment; git commits after the due date will cause the assignment to be considered late.

R Markdown documents included as part of your solutions must not install packages, and should only load the packages necessary for your submission to knit.

Points

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 structure of your submission, especially the use git and GitHub for reproducibility, R Projects to organize your work, R Markdown to write reproducible reports, relative paths to load data from local files, and reasonable naming structures for your files.

To that end:

- create a public GitHub repo + local R Project; I suggest naming this repo / directory bios731_hw1_YourLastName (e.g. bios731_hw1_wrobel for Julia)
- Submit your whole project folder to GitHub
- Submit a PDF knitted from Rmd to Canvas. Your solutions to the problem here should be implemented in your .Rmd file, and your git commit history should reflect the process you used to solve these Problems.

Problem 1

Simulation study: our goal in this homework will be to plan a well-organized simulation study for multiple linear regression and bootstrapped confidence intervals.

Below is a multiple linear regression model, where we are interested in primarily treatment effect.

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

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 $\mathbf{Z_i}$
- $\boldsymbol{\gamma}$: vector of regression coefficient values for confounders
- ϵ_i: errors, we will vary how these are defined

In our simulation, we want to

- Estimate $\beta_{treatment}$ and $se(\hat{\beta}_{treatment})$
  - Evaluate $\beta_{treatment}$ through bias and coverage
  - We will use 3 methods to compute $se(\hat{\beta}_{treatment})$ and coverage:
    1. Wald confidence intervals (the standard approach)
    2. Nonparametric bootstrap percentile intervals
    3. Nonparametric bootstrap t intervals
  - Evaluate computation times for each method to compute a confidence interval
- Evaluate these properties at:
  - Sample size $n \in \{10, 50, 500\}$
  - True values $\beta_{treatment} \in \{0, 0.5, 2\}$
  - True ϵ_i normally distributed with $\epsilon_i \sim N(0, 2)$
  - True ϵ_i coming from a right skewed distribution
    - **Hint**: try $\epsilon_i \sim logNormal(0, \log(2))$
- Assume that there are no confounders ($\boldsymbol{\gamma} = 0$)
- Use a full factorial design

Problem 1.1 ADEMP Structure

Answer the following questions:

- How many simulation scenarios will you be running?

Considering there are 3 sample sizes, 3 true values, 2 random error distributions, there would be *18* simulation scenarios.

- What are the estimand(s)

The estimands include $\beta_{treatment}$ and $se(\hat{\beta}_{treatment})$

- What method(s) are being evaluated/compared?

3 methods ar used to compute $se(\hat{\beta}_{treatment})$ and coverage.

- What are the performance measure(s)?

Bias for $\beta_{treatment}$, $se(\hat{\beta}_{treatment})$ and coverage measured under three methods and computation time.

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 we perform for each simulation scenario? Implement this number of simulations throughout your simulation study.

```r
0.95*(1-0.95)/0.01^2
```

```
## [1] 475
```

Therefore, we need to perform *475* simulation for each scenario.

Problem 1.3 Implementation

We will execute this full simulation study. For full credit, make sure to implement the following:

- Well structured scripts and subfolders following guidance from `project_organization` lecture
- Use relative file paths to access intermediate scripts and data objects
- Use readable code practices
- Parallelize your simulation scenarios
- Save results from each simulation scenario in an intermediate `.Rda` or `.rds` dataset in a `data` subfolder
  - Ignore these data files in your `.gitignore` file so when pushing and committing to GitHub they don't get pushed to remote
- Make sure your folder contains a Readme explaining the workflow of your simulation study
  - should include how files are executed and in what order
- Ensure reproducibility! I should be able to clone your GitHub repo, open your `.Rproj` file, and run your simulation study

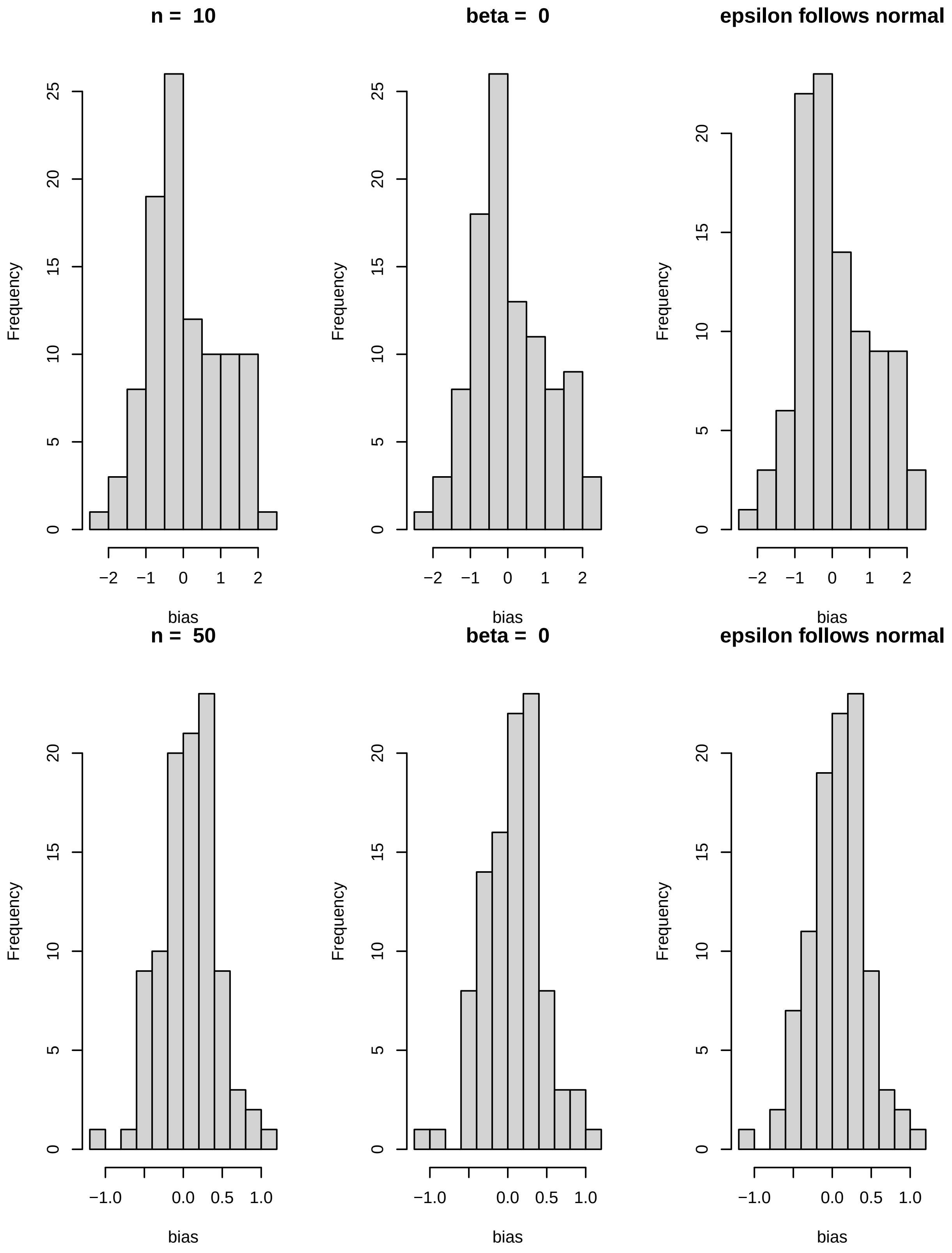
Problem 1.4 Results summary

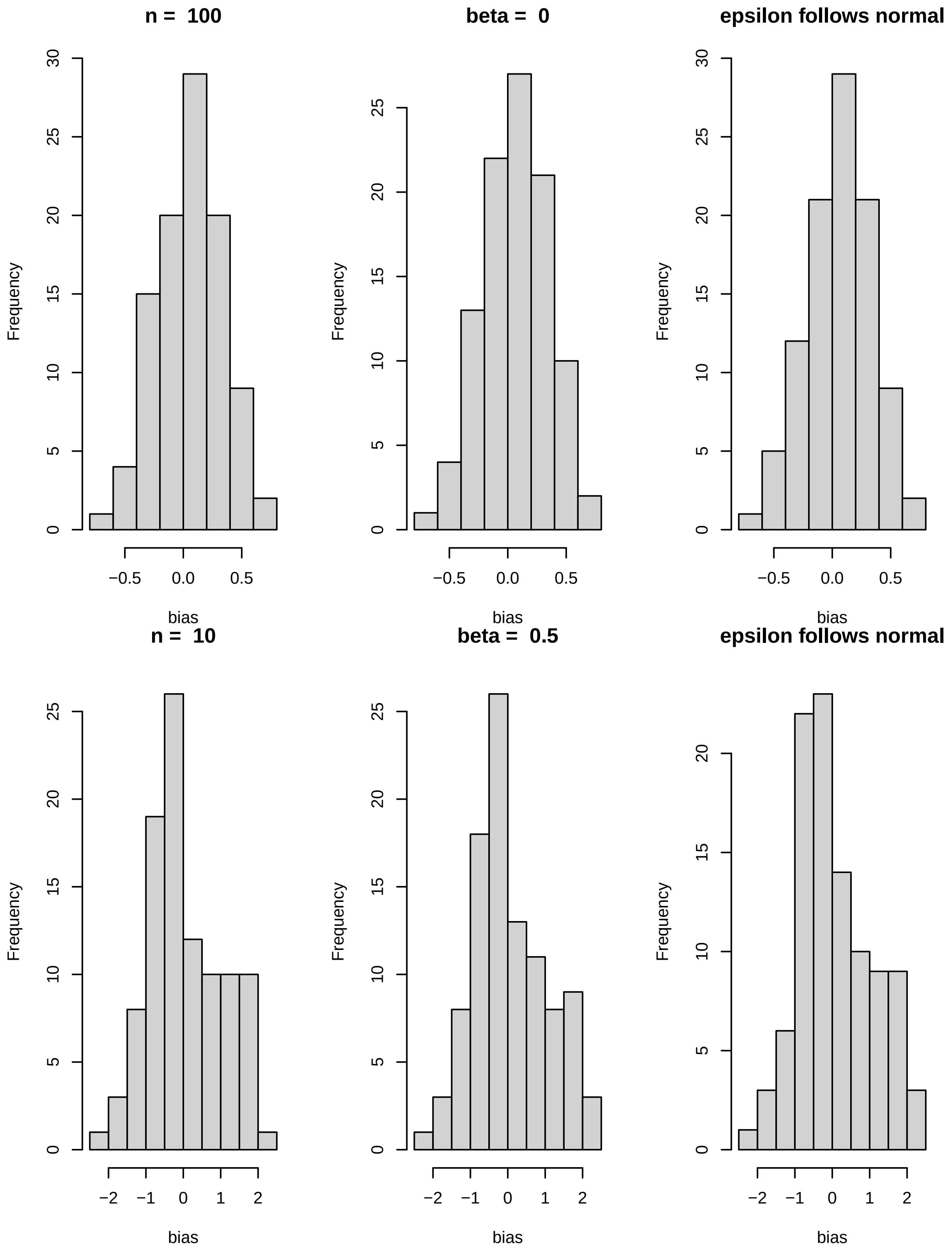
Create a plot or table to summarize simulation results across scenarios and methods for each of the following.

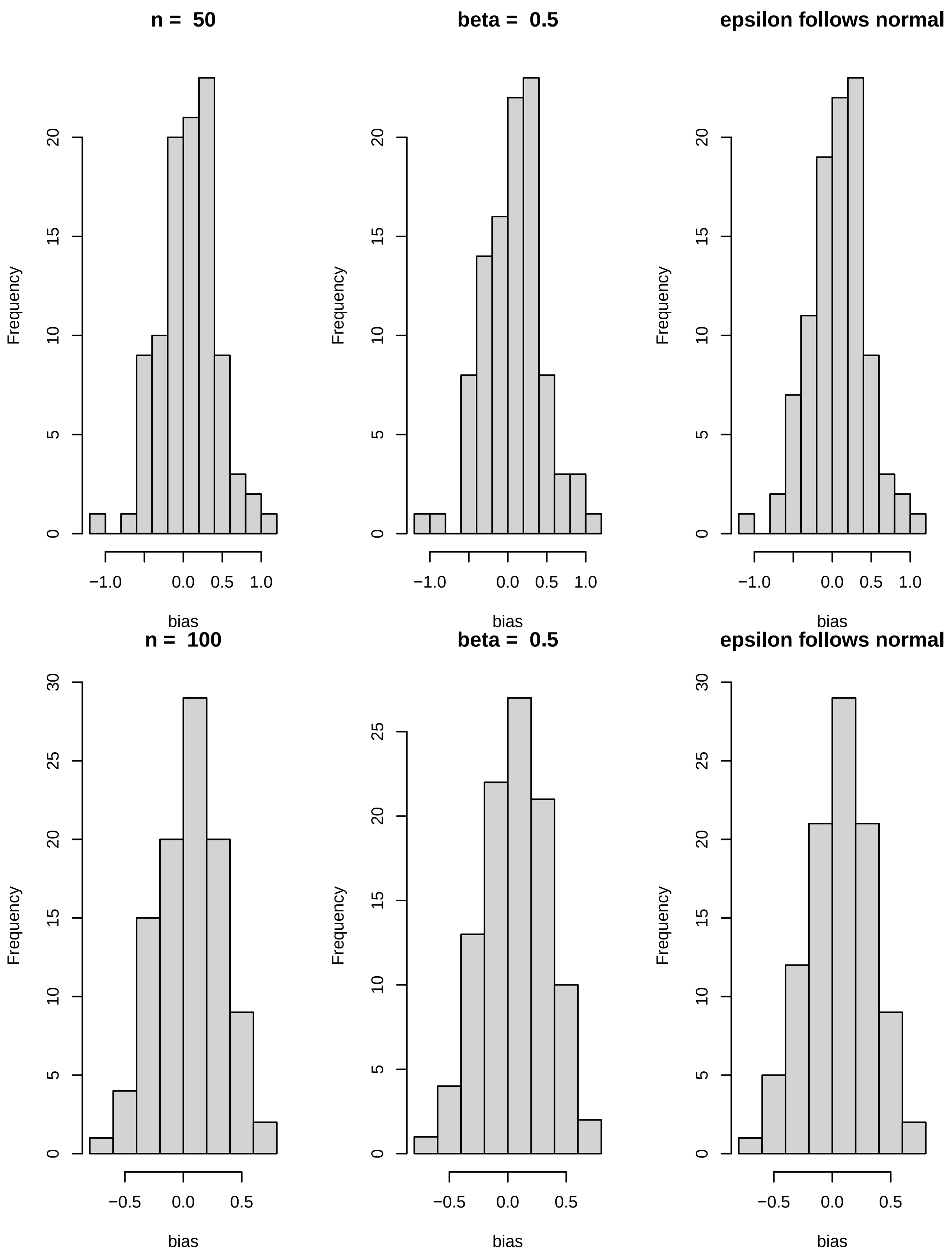
```r
n = c(10, 50, 100)
beta_true = c(0, 0.5, 2)
eps_dist = c("normal", "log-normal")

params = expand.grid(n = n,
                     beta_true = beta_true,
                     eps_dist = eps_dist)
```

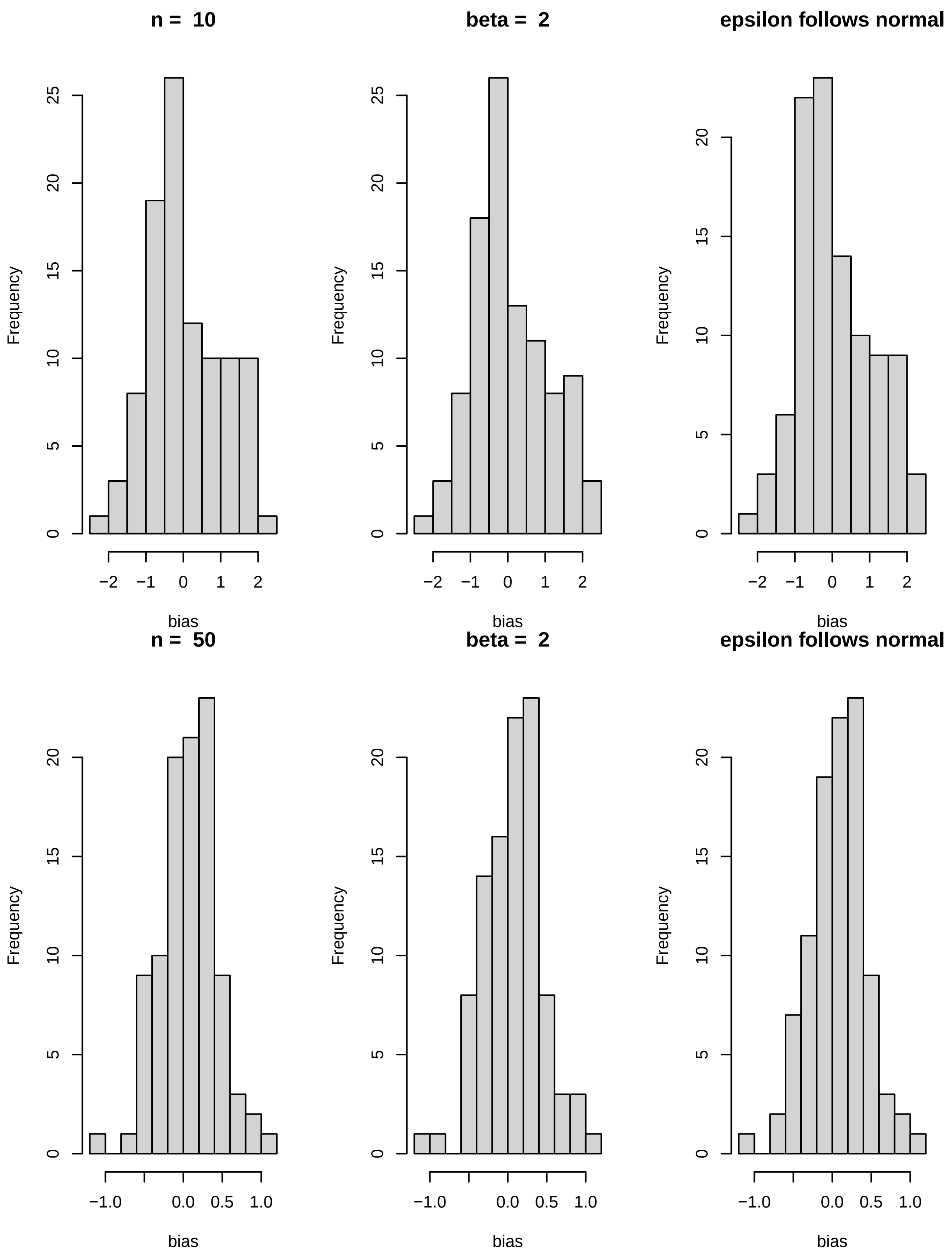
- Bias of $\hat{\beta}$

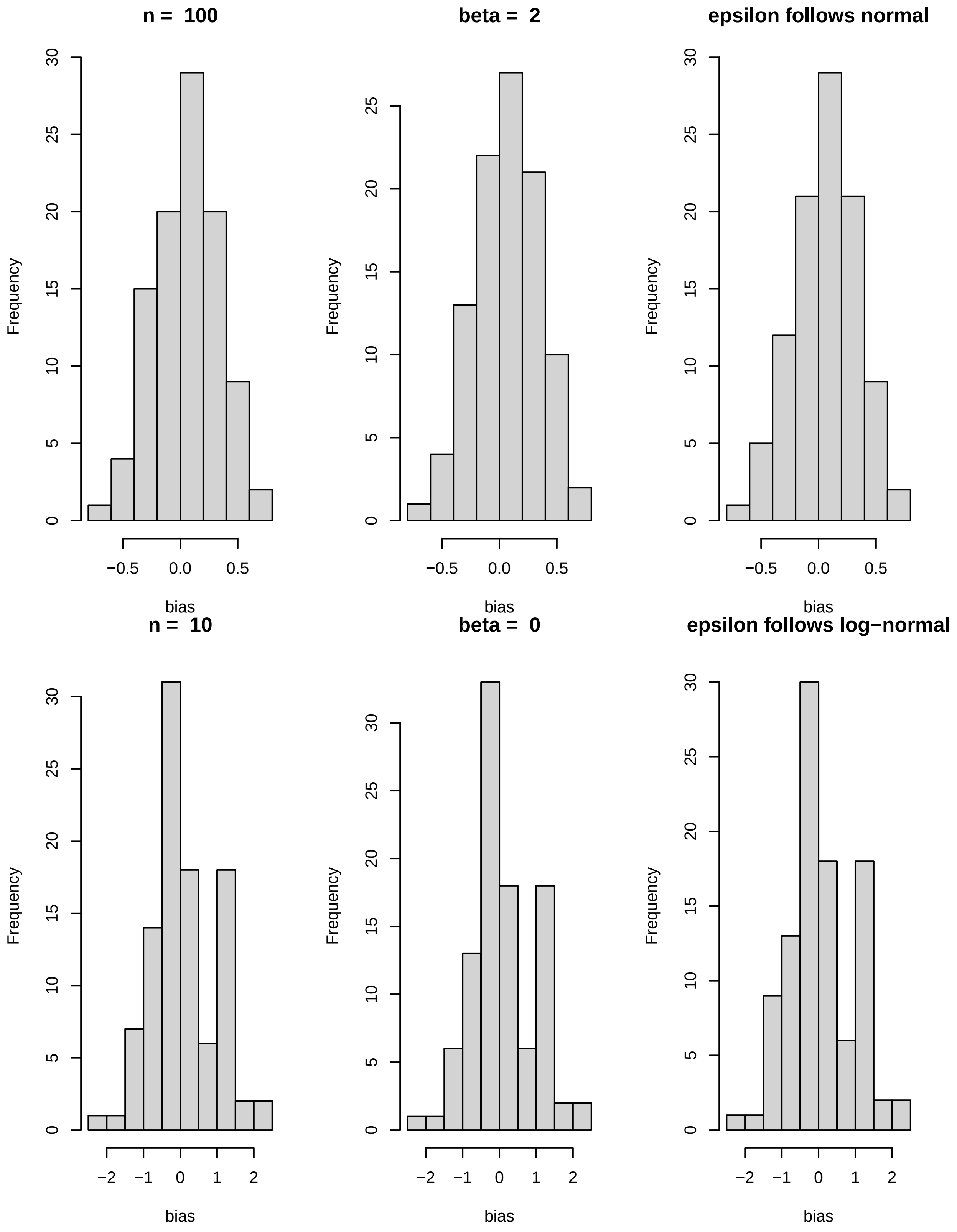
```r
par(mfrow = c(1,3))
for (i in 1:18){
  filename = paste0("scenario_", i, ".RDA")
  load(here::here("data", filename))
  hist(sapply(results, `[`, 1), xlab = "bias", main = paste("n = ", params[i,1]))
  hist(sapply(results, `[`, 2), xlab = "bias", main = paste("beta = ", params[i,2]))
  hist(sapply(results, `[`, 3), xlab = "bias", main = paste("epsilon follows", params[i,3]))
}
```

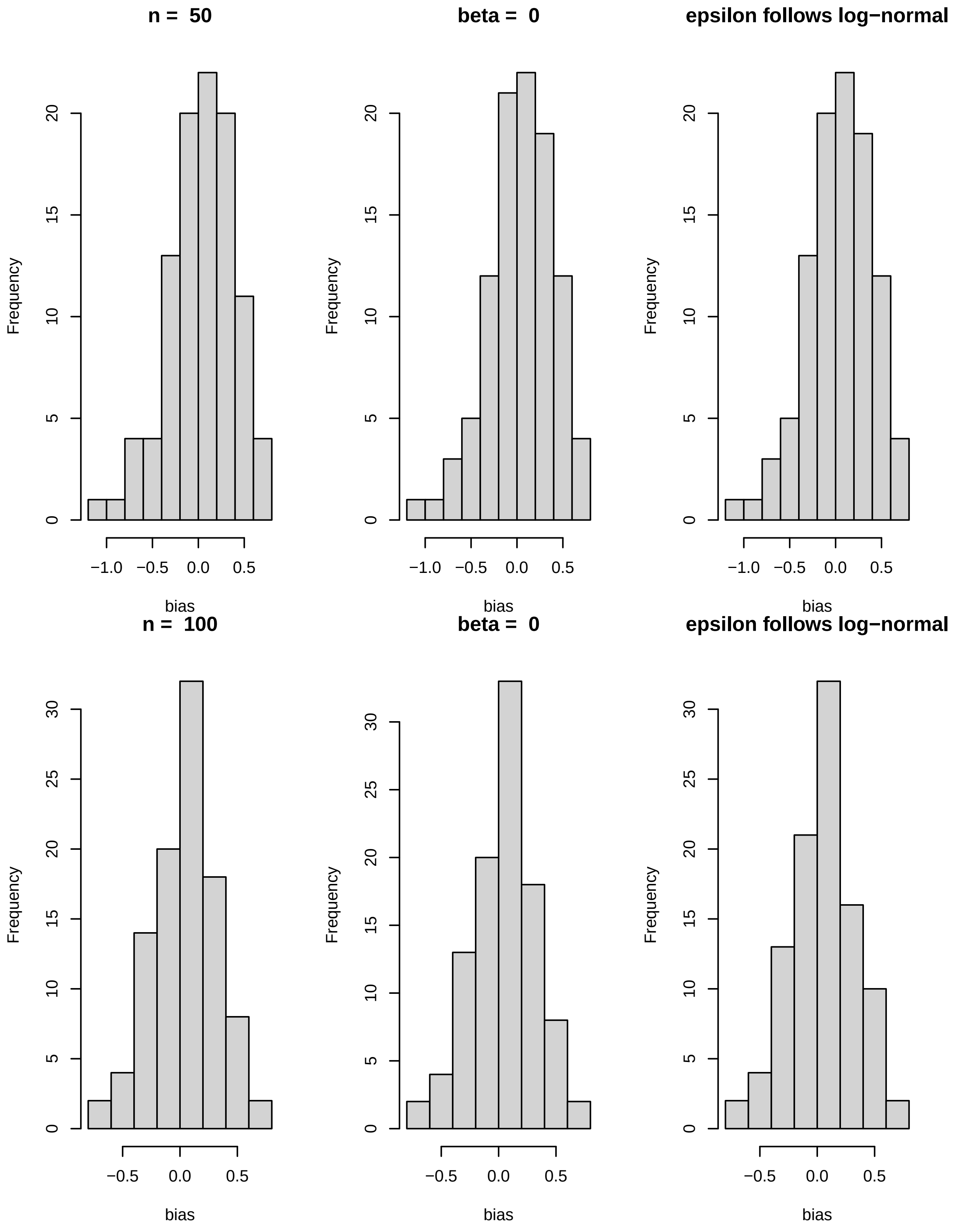

n = 10
beta = 0
epsilon follows normal
Frequency
bias
n = 50
beta = 0
epsilon follows normal
Frequency
bias

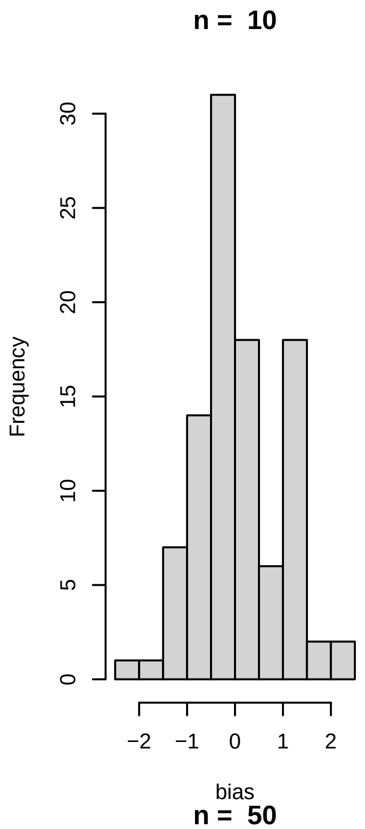

n = 100
beta = 0
epsilon follows normal
n = 10
beta = 0.5
epsilon follows normal
Frequency
bias
0
5
10
15
20
25
30
−0.5
0.0
0.5
−2
−1
0
1
2

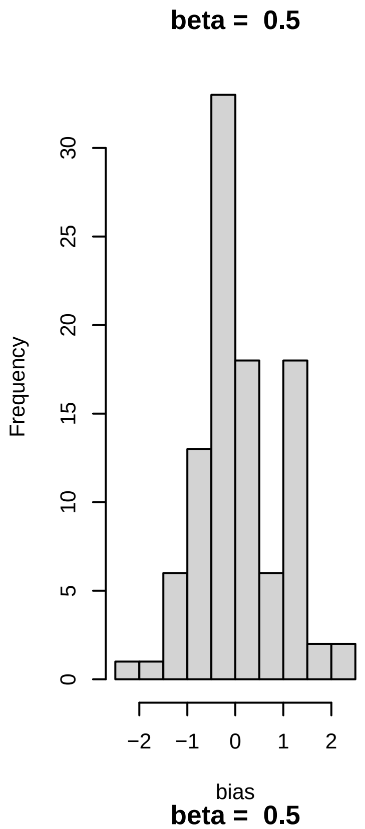

n = 50
beta = 0.5
epsilon follows normal
n = 100
beta = 0.5
epsilon follows normal
Frequency
bias

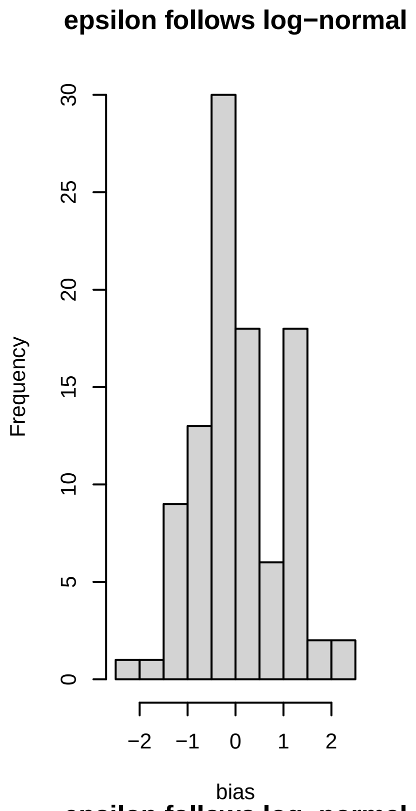

n = 10
beta = 2
epsilon follows normal
n = 50
beta = 2
epsilon follows normal
Frequency
bias
0
5
10
15
20
25
−2
−1
0
1
2
−1.0
0.0
0.5
1.0

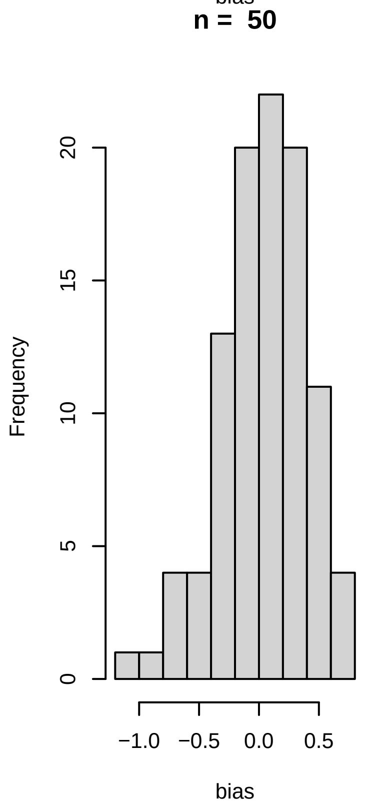

n = 100
beta = 2
epsilon follows normal
n = 10
beta = 0
epsilon follows log−normal
Frequency
bias

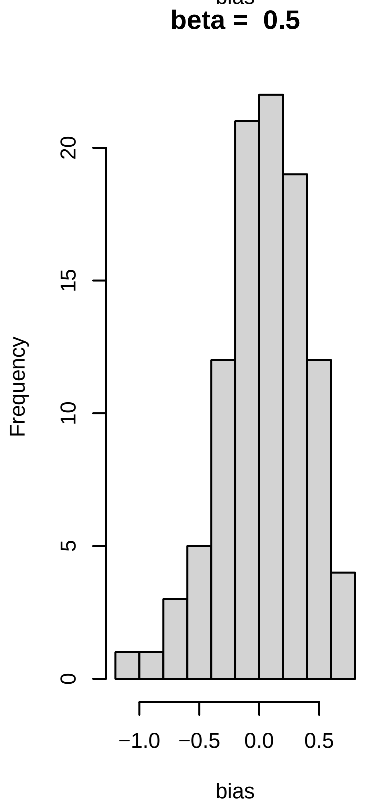

n = 50
beta = 0
epsilon follows log−normal
Frequency
bias
n = 100
beta = 0
epsilon follows log−normal
Frequency
bias

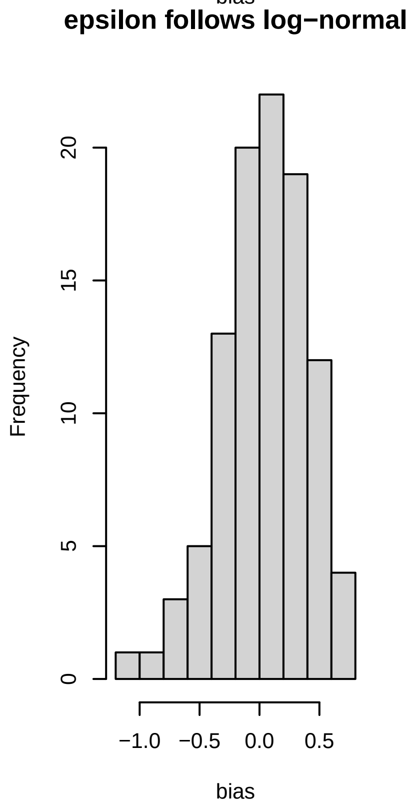

n = 10
Frequency
bias

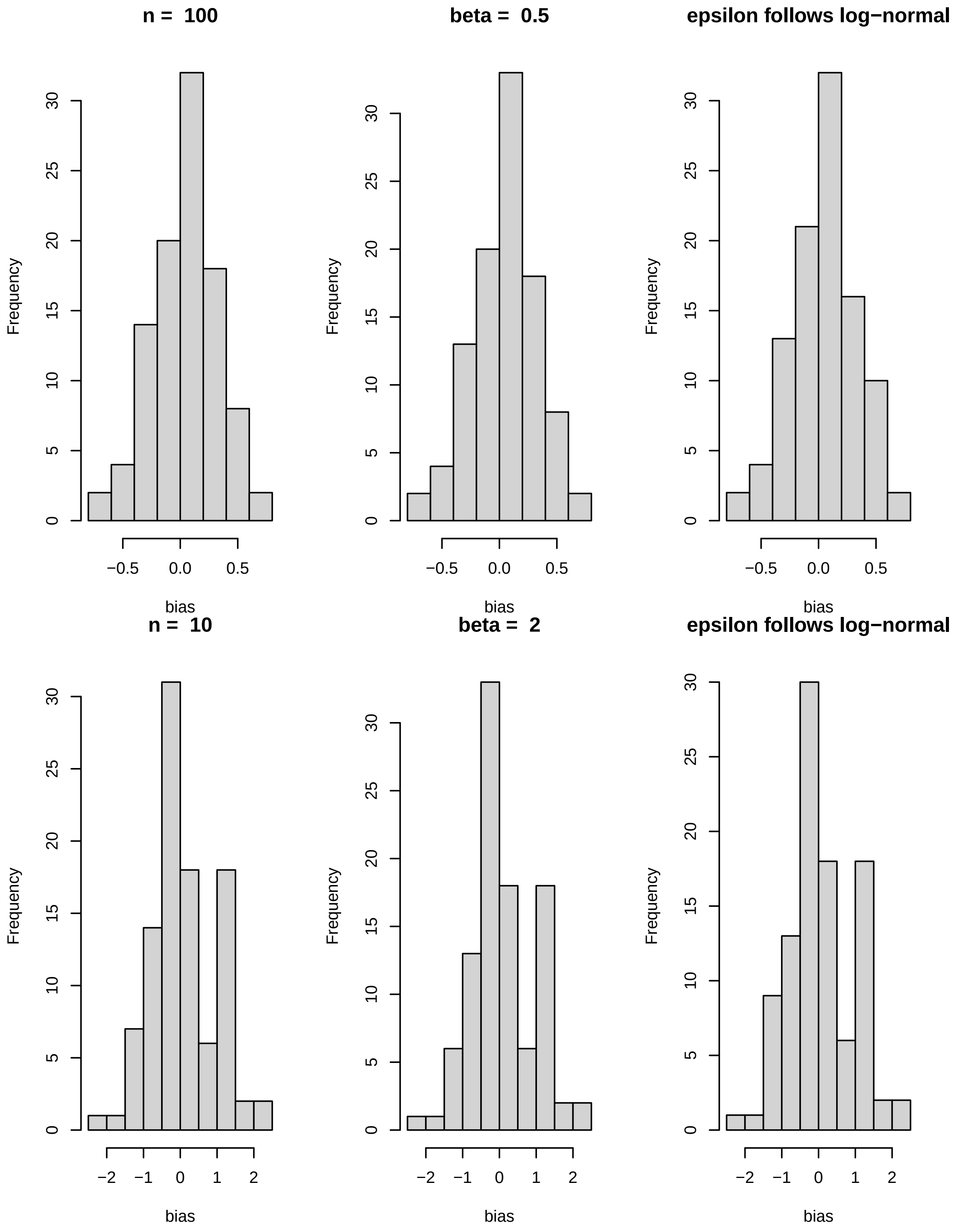

beta = 0.5
Frequency
bias

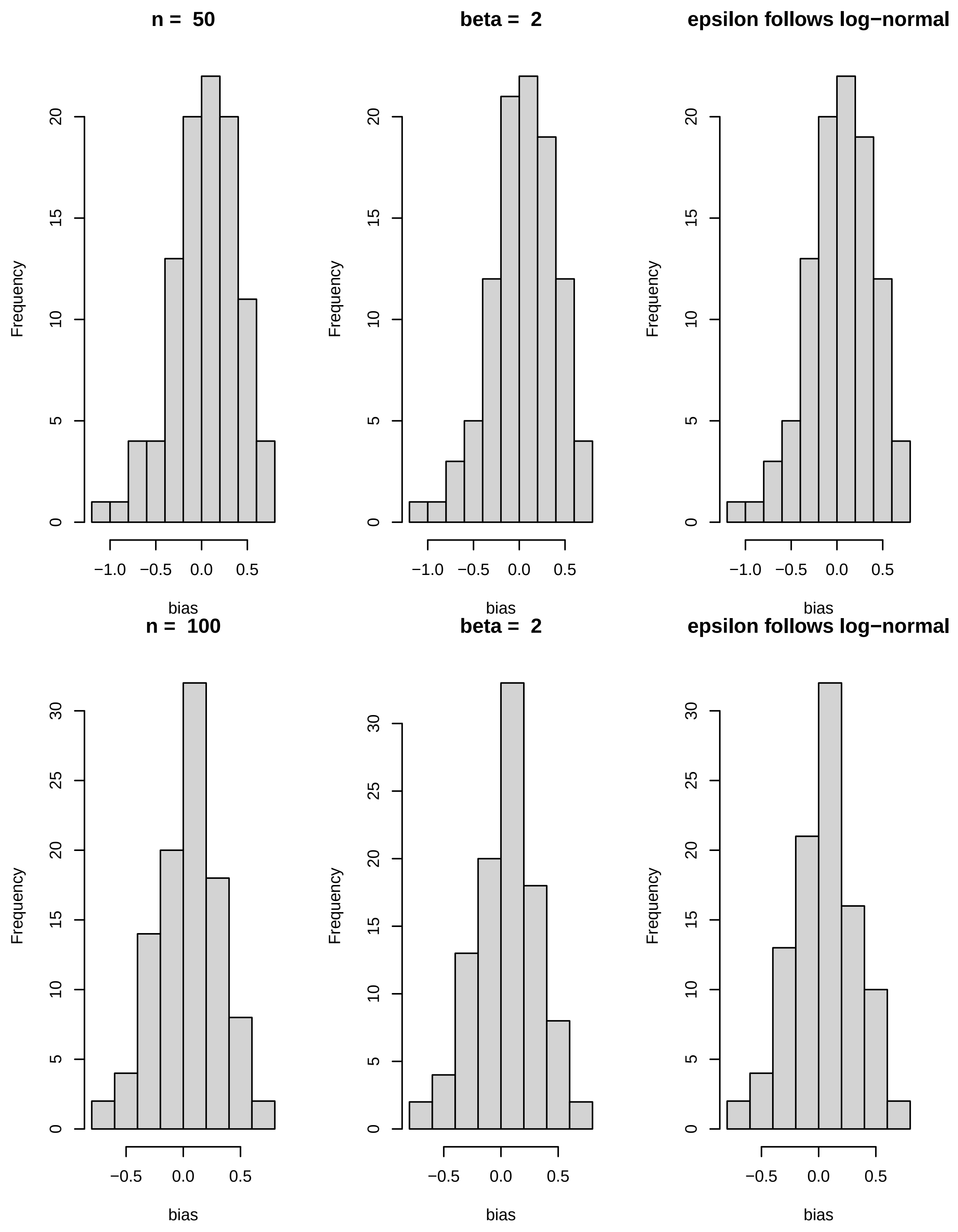

epsilon follows log−normal
Frequency
bias


n = 50
Frequency
bias


beta = 0.5
Frequency
bias


epsilon follows log−normal
Frequency
bias


n = 100
beta = 0.5
epsilon follows log−normal
n = 10
beta = 2
epsilon follows log−normal
Frequency
bias



- Coverage of $\hat{\beta}$

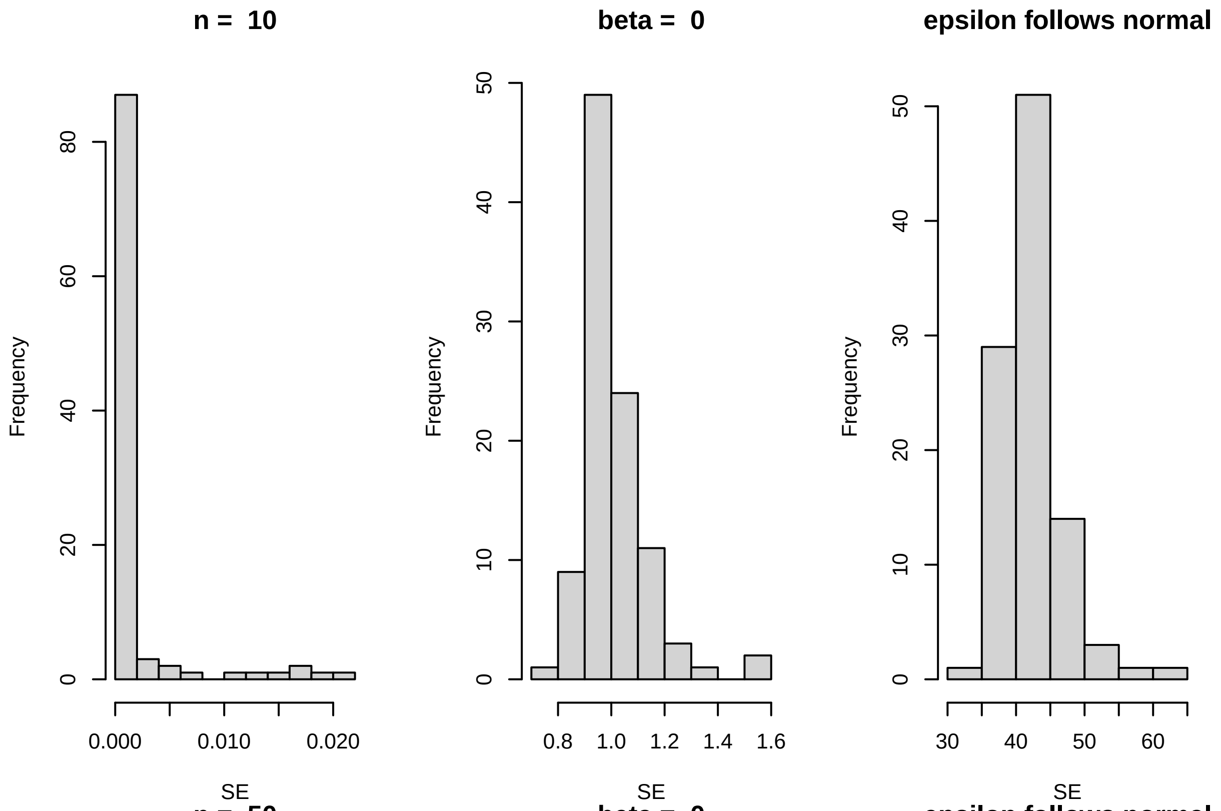
```
coverage_table = params
for (i in 1:18){
```

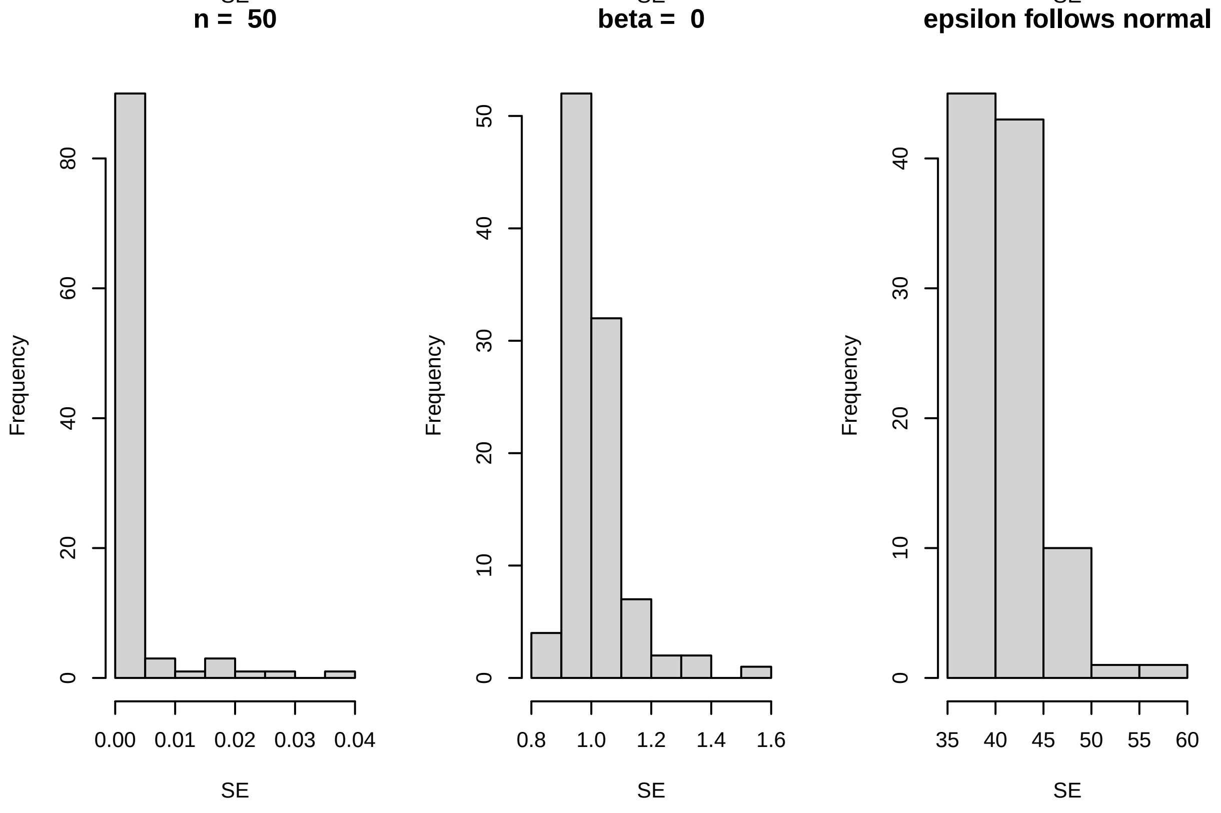
```r
  filename = paste0("scenario_", i, ".RDA")
  load(here::here("data", filename))
  coverage_table$Wald[i] = mean(sapply(results, `[`, 7))
  coverage_table$boot_np[i] = mean(sapply(results, `[`, 8))
  coverage_table$boot_t[i] = mean(sapply(results, `[`, 9))
}
print(coverage_table)
```

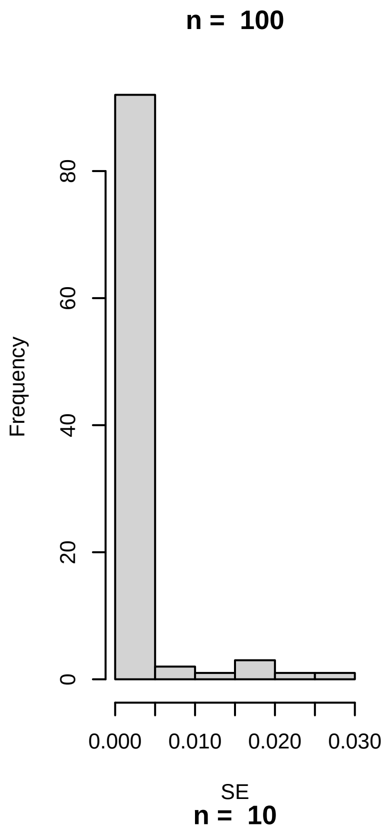
```
##      n beta_true   eps_dist Wald boot_np boot_t
## 1   10       0.0     normal 0.94    0.84   0.96
## 2   50       0.0     normal 0.94    0.94   0.95
## 3  100       0.0     normal 0.94    0.96   0.95
## 4   10       0.5     normal 0.94    0.84   0.96
## 5   50       0.5     normal 0.94    0.94   0.95
## 6  100       0.5     normal 0.94    0.96   0.95
## 7   10       2.0     normal 0.94    0.84   0.96
## 8   50       2.0     normal 0.94    0.94   0.95
## 9  100       2.0     normal 0.94    0.96   0.95
## 10  10       0.0 log-normal 0.94    0.87   0.93
## 11  50       0.0 log-normal 0.92    0.93   0.91
## 12 100       0.0 log-normal 0.97    0.93   0.92
## 13  10       0.5 log-normal 0.94    0.87   0.93
## 14  50       0.5 log-normal 0.92    0.93   0.91
## 15 100       0.5 log-normal 0.97    0.93   0.92
## 16  10       2.0 log-normal 0.94    0.87   0.93
## 17  50       2.0 log-normal 0.92    0.93   0.91
## 18 100       2.0 log-normal 0.97    0.93   0.92
```

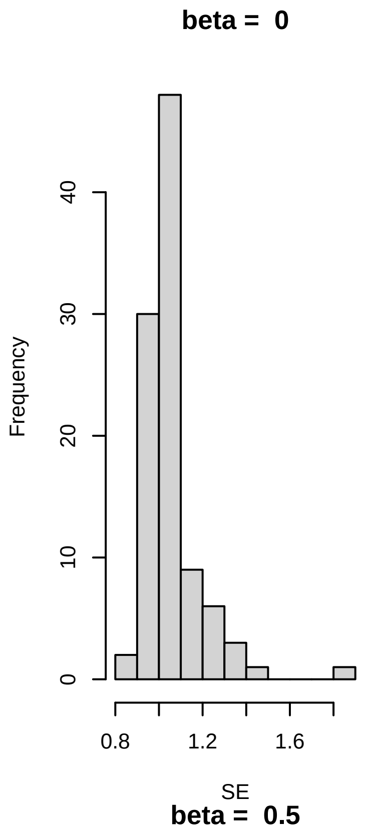
- Distribution of $se(\hat{\beta})$

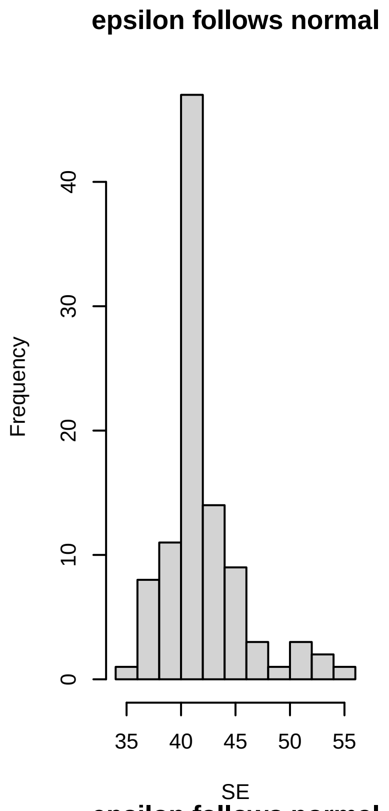
```r
par(mfrow = c(1,3))
for (i in 1:18){
  filename = paste0("scenario_", i, ".RDA")
  load(here::here("data", filename))
  hist(sapply(results, `[`, 10), xlab = "SE", main = paste("n = ", params[i,1]))
  hist(sapply(results, `[`, 11), xlab = "SE", main = paste("beta = ", params[i,2]))
  hist(sapply(results, `[`, 12), xlab = "SE", main = paste("epsilon follows", params[i,3]))
}
```

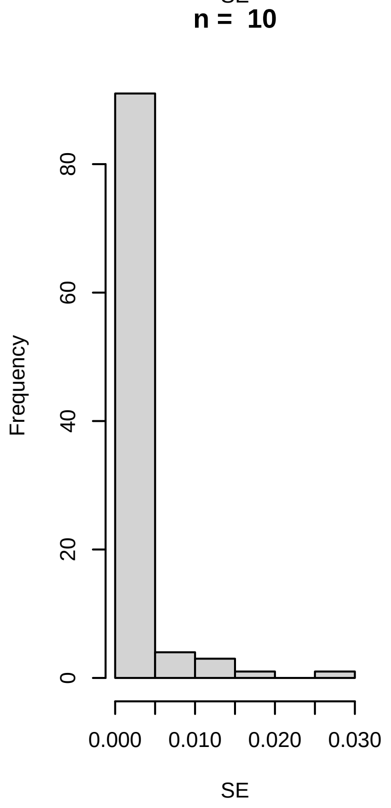

n = 10
beta = 0
epsilon follows normal
Frequency
SE

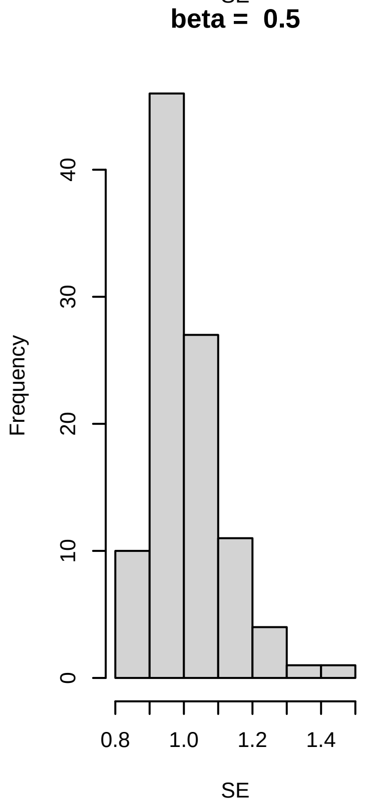

n = 50
beta = 0
epsilon follows normal
Frequency
SE

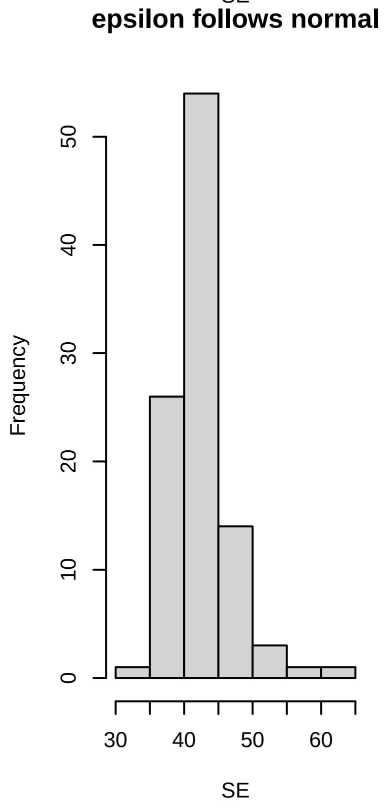

n = 100
Frequency
SE
0
20
40
60
80
0.000
0.010
0.020
0.030

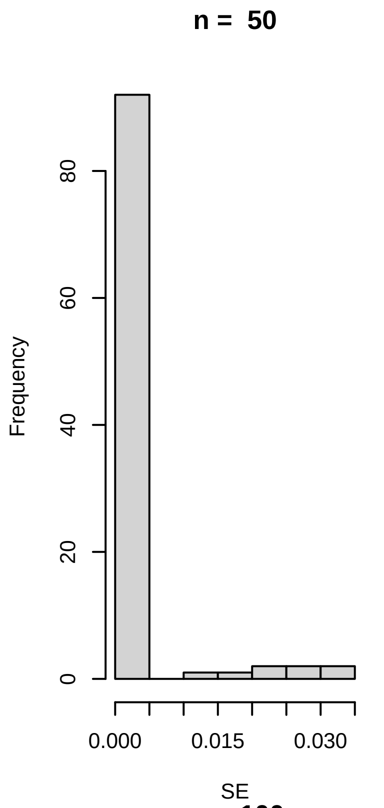

beta = 0
Frequency
SE
0
10
20
30
40
0.8
1.2
1.6

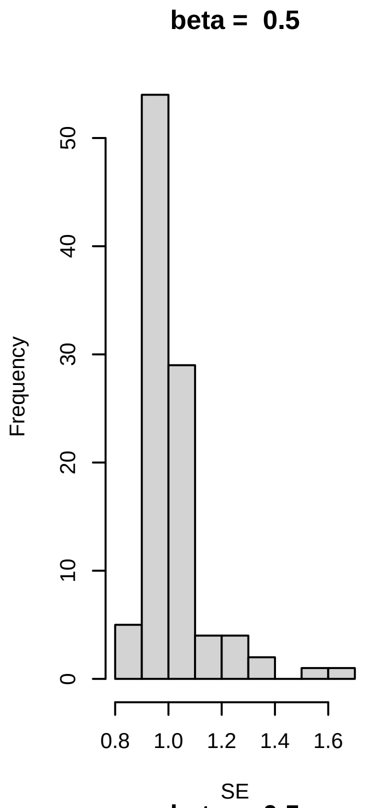

epsilon follows normal
Frequency
SE
0
10
20
30
40
35
40
45
50
55

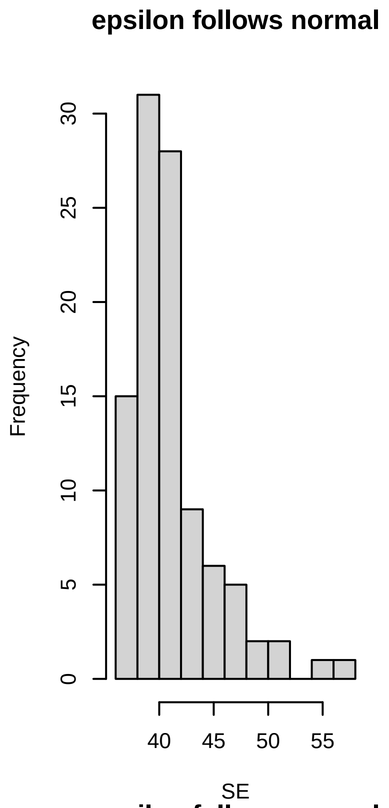

n = 10
Frequency
SE
0
20
40
60
80
0.000
0.010
0.020
0.030

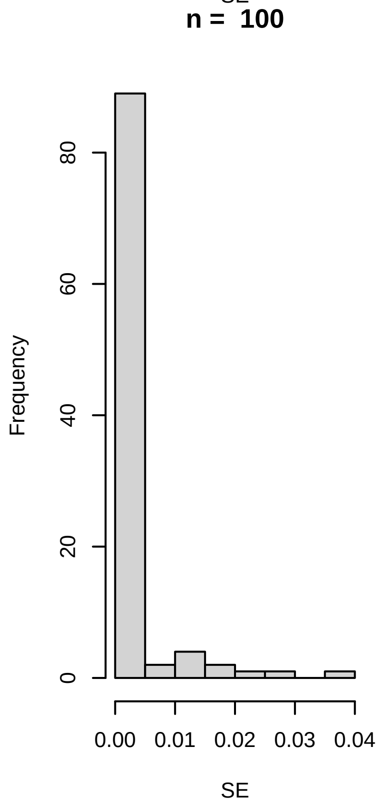

beta = 0.5
Frequency
SE
0
10
20
30
40
0.8
1.0
1.2
1.4

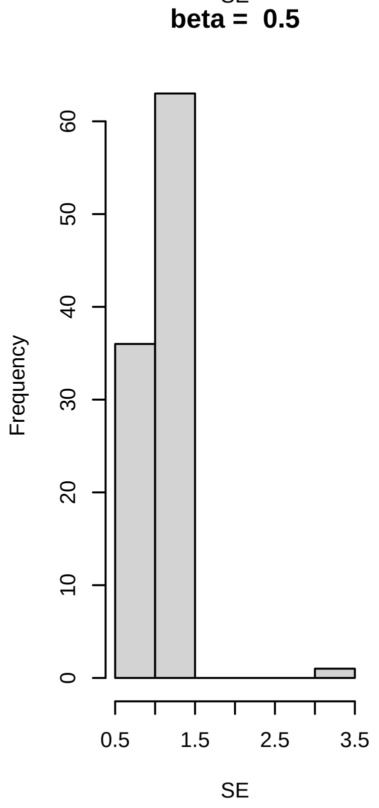

epsilon follows normal
Frequency
SE
0
10
20
30
40
50
30
40
50
60

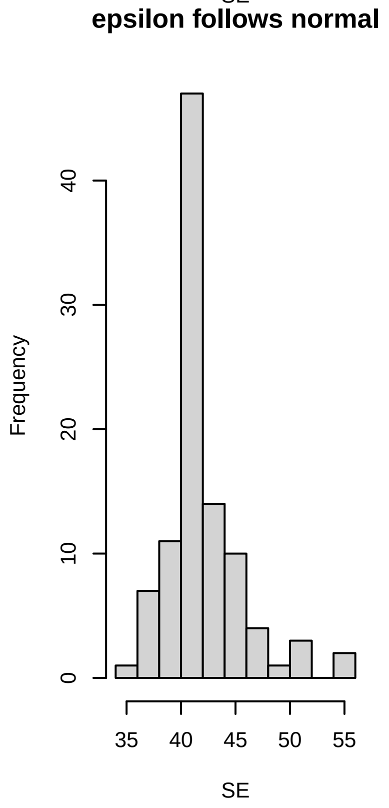

n = 50
Frequency
0
20
40
60
80
0.000
0.015
0.030
SE

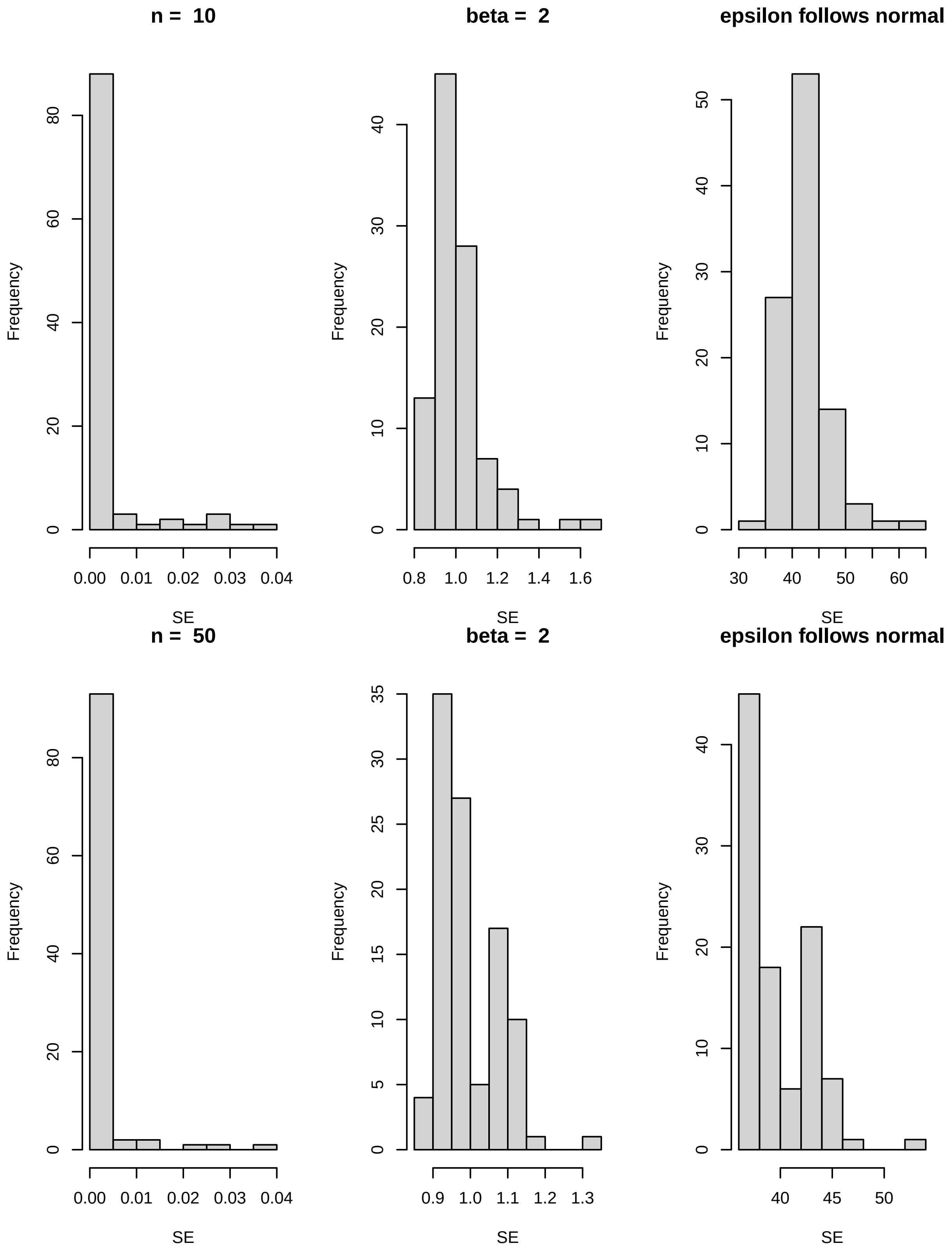

beta = 0.5
Frequency
0
10
20
30
40
50
0.8
1.0
1.2
1.4
1.6
SE


epsilon follows normal
Frequency
0
5
10
15
20
25
30
40
45
50
55
SE

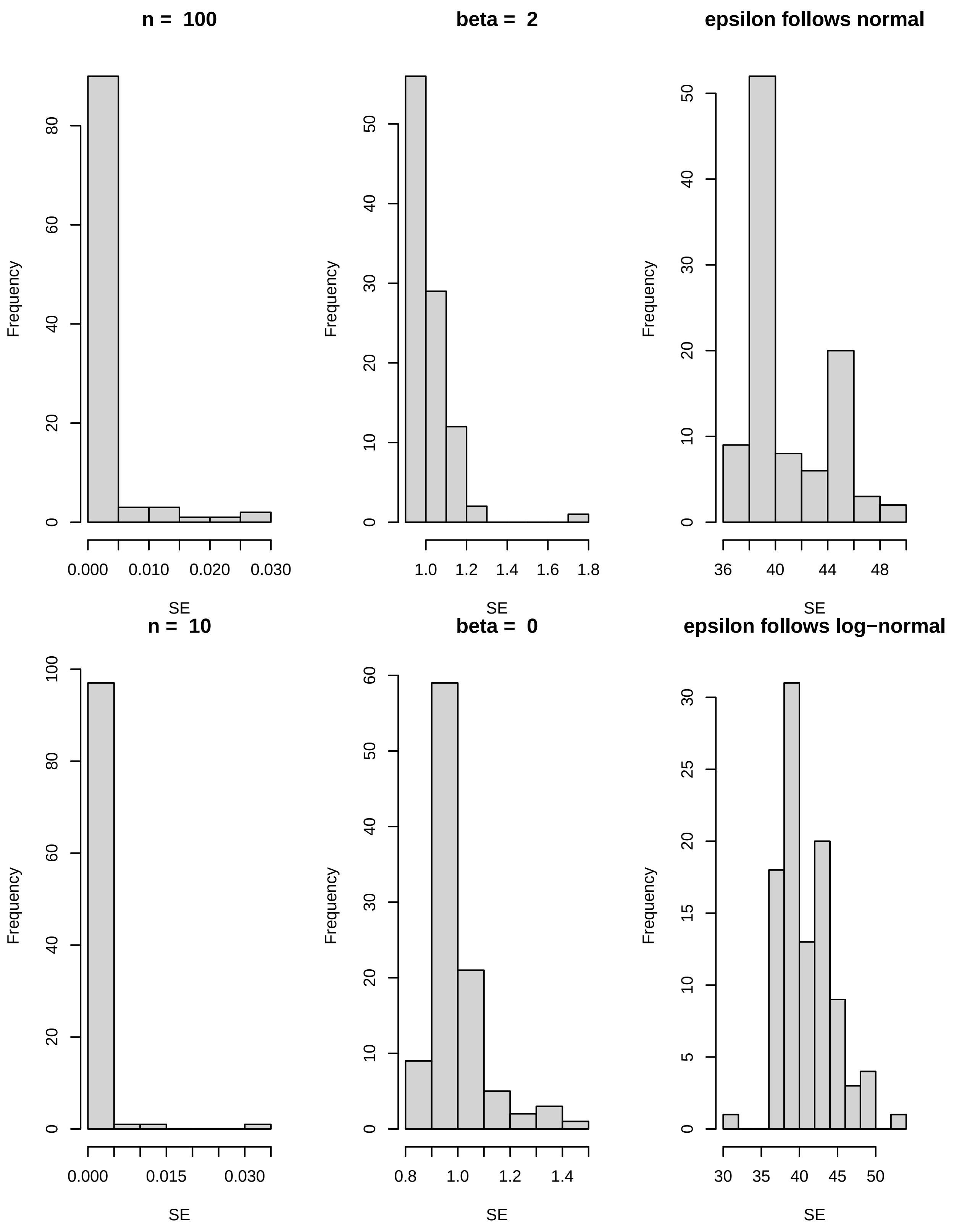

n = 100
Frequency
0
20
40
60
80
0.00
0.01
0.02
0.03
0.04
SE

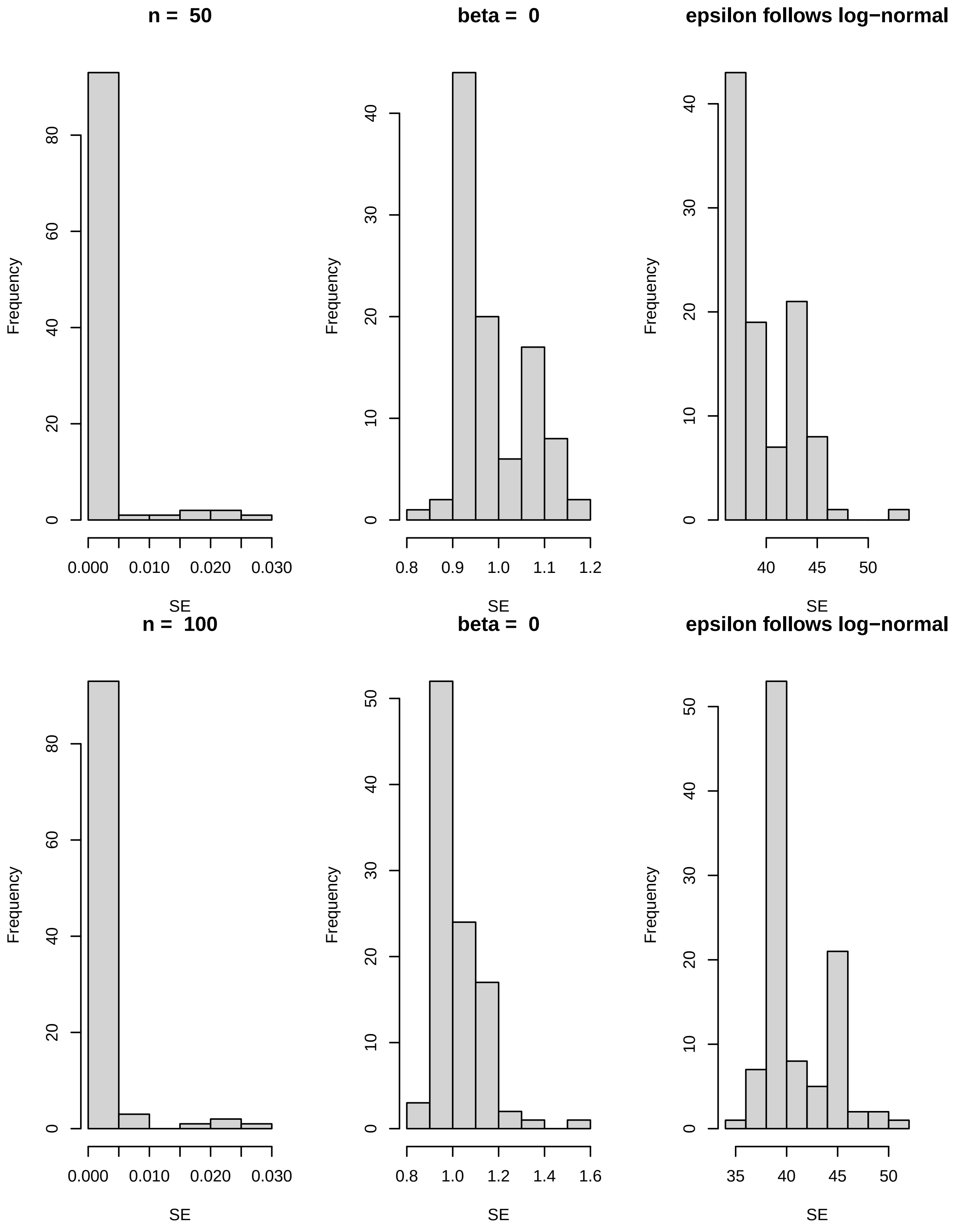

beta = 0.5
Frequency
0
10
20
30
40
50
60
0.5
1.5
2.5
3.5
SE

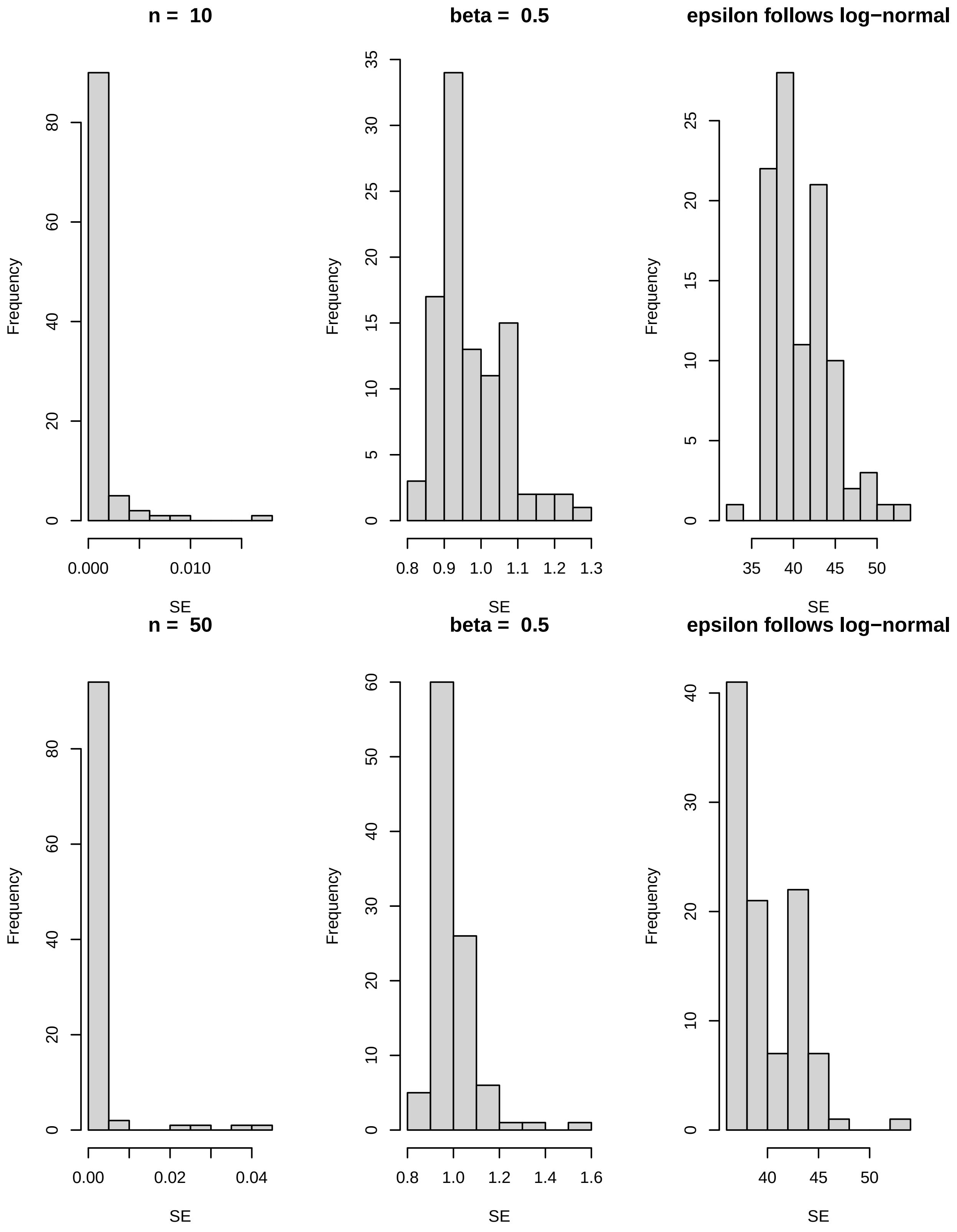

epsilon follows normal
Frequency
0
10
20
30
40
35
40
45
50
55
SE



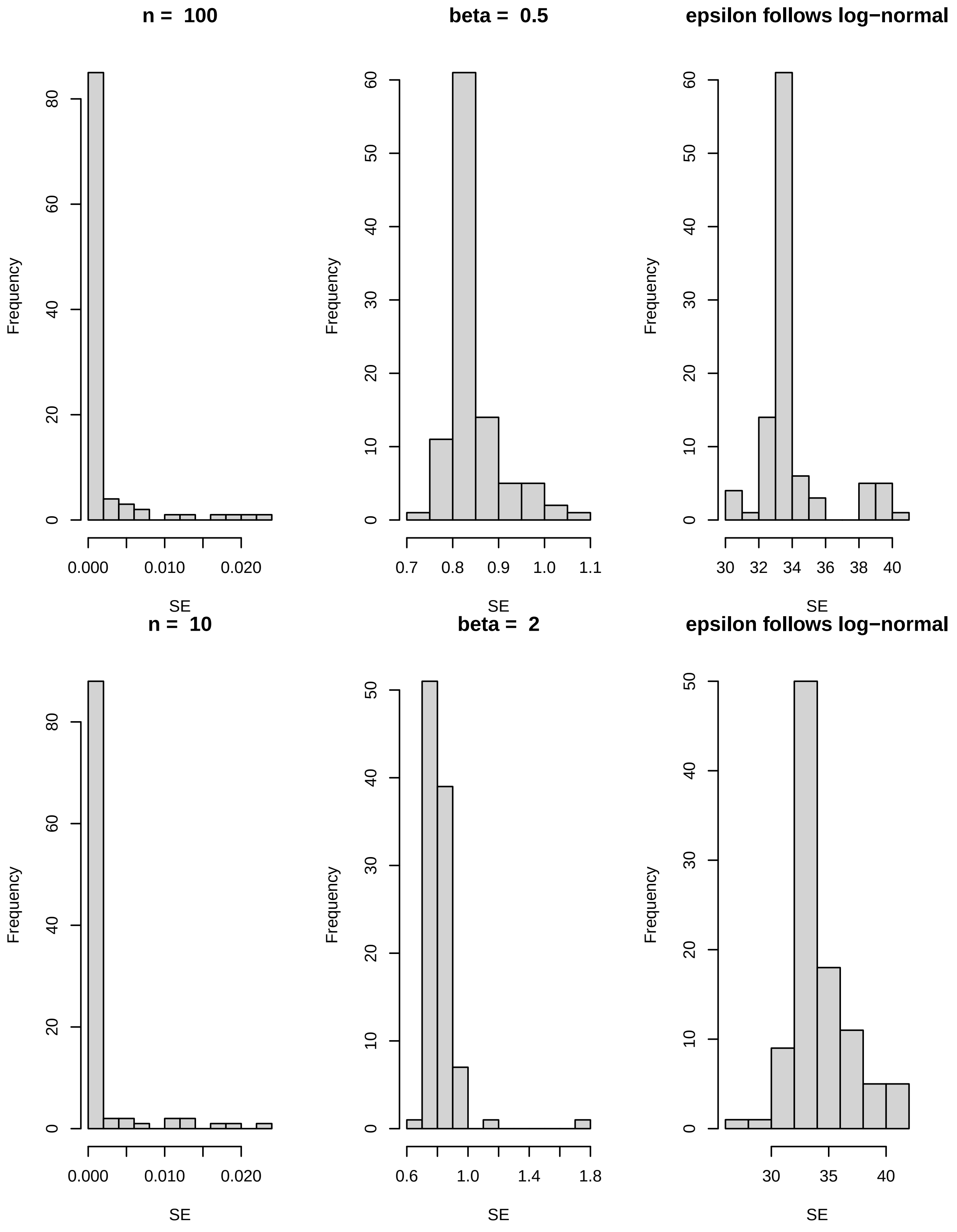
n = 10
beta = 2
epsilon follows normal
n = 50
beta = 2
epsilon follows normal
Frequency
SE

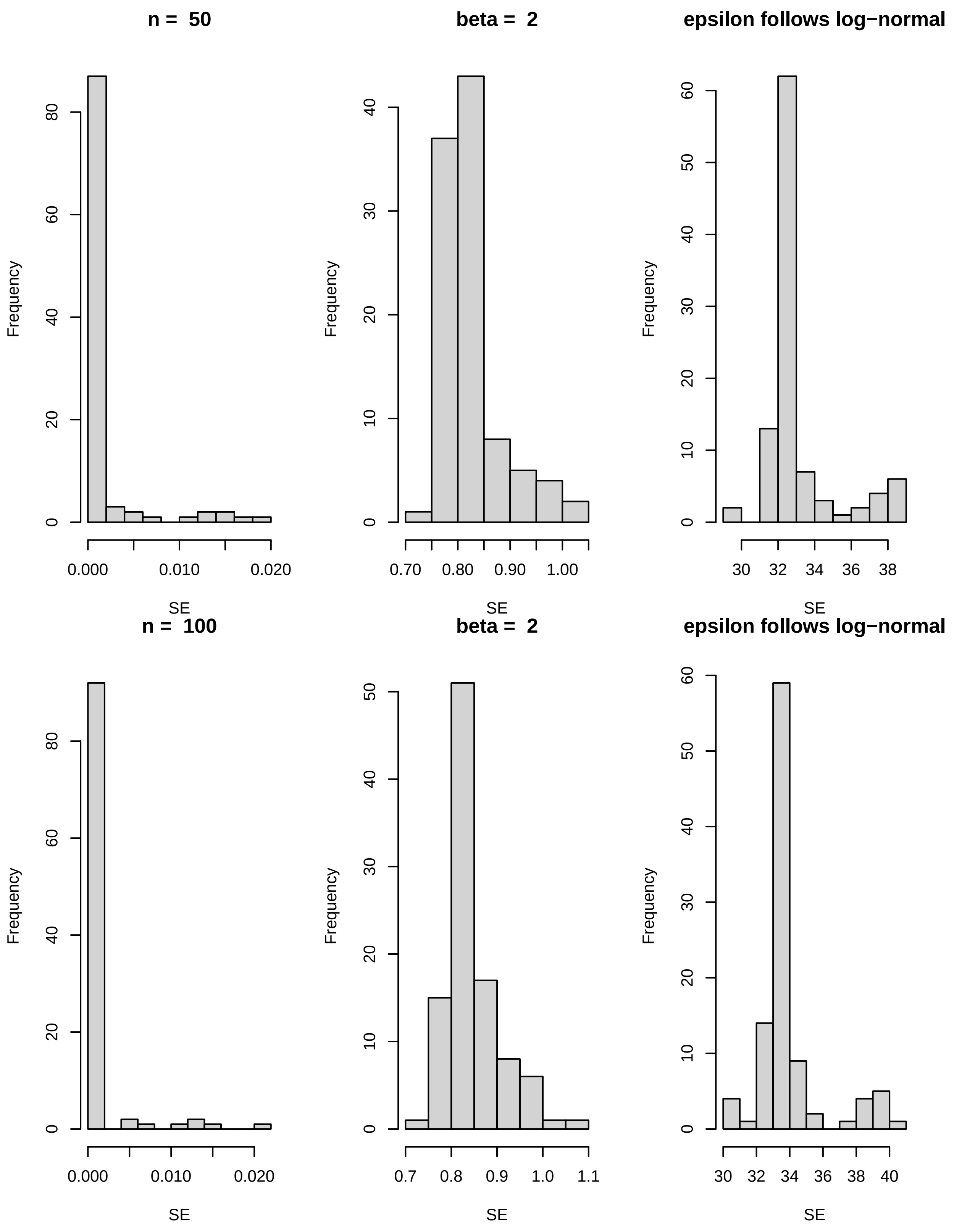

n = 100
Frequency
0
20
40
60
80
0.000
0.010
0.020
0.030
SE
beta = 2
Frequency
0
10
20
30
40
50
1.0
1.2
1.4
1.6
1.8
SE
epsilon follows normal
Frequency
0
10
20
30
40
50
36
40
44
48
SE
n = 10
Frequency
0
20
40
60
80
100
0.000
0.015
0.030
SE
beta = 0
Frequency
0
10
20
30
40
50
60
0.8
1.0
1.2
1.4
SE
epsilon follows log−normal
Frequency
0
5
10
15
20
25
30
30
35
40
45
50
SE


n = 50
beta = 0
epsilon follows log−normal
Frequency
SE
n = 100
beta = 0
epsilon follows log−normal
Frequency
SE



n = 10
beta = 0.5
epsilon follows log−normal
n = 50
beta = 0.5
epsilon follows log−normal
Frequency
SE


n = 100
beta = 0.5
epsilon follows log−normal
n = 10
beta = 2
epsilon follows log−normal
Frequency
SE



- Computation time across methods

```
time_table = params
for (i in 1:18){
```

```r
  filename = paste0("scenario_", i, ".RDA")
  load(here::here("data", filename))
  time_table$Wald[i] = mean(sapply(results, `[`, 10))
  time_table$boot_np[i] = mean(sapply(results, `[`, 11))
  time_table$boot_t[i] = mean(sapply(results, `[`, 12))
}
print(time_table)
```

```
##      n beta_true   eps_dist        Wald   boot_np   boot_t
## 1   10       0.0     normal 0.002194178 1.0134367 42.25628
## 2   50       0.0     normal 0.002761495 1.0146422 41.05499
## 3  100       0.0     normal 0.002377207 1.0487651 42.05240
## 4   10       0.5     normal 0.002058105 1.0077273 42.26493
## 5   50       0.5     normal 0.002906024 1.0153047 41.10562
## 6  100       0.5     normal 0.002874589 1.0659388 42.04530
## 7   10       2.0     normal 0.003416040 1.0141745 42.25394
## 8   50       2.0     normal 0.002303278 0.9926850 39.89224
## 9  100       2.0     normal 0.002572837 1.0154210 40.79637
## 10  10       0.0 log-normal 0.001599548 0.9875447 41.01721
## 11  50       0.0 log-normal 0.002333846 0.9853371 39.94453
## 12 100       0.0 log-normal 0.002187076 1.0214936 40.83561
## 13  10       0.5 log-normal 0.001435907 0.9724436 41.00194
## 14  50       0.5 log-normal 0.002389643 0.9970392 39.94585
## 15 100       0.5 log-normal 0.002273593 0.8467163 33.96239
## 16  10       2.0 log-normal 0.002139938 0.8231112 34.10805
## 17  50       2.0 log-normal 0.002092187 0.8272930 33.13036
## 18 100       2.0 log-normal 0.001858473 0.8486792 33.98506
```

If creating a plot, I encourage faceting. Include informative captions for each plot and/or table.

Problem 1.5 Discussion

Interpret the results summarized in Problem 1.4. First, write a **paragraph** summarizing the main findings of your simulation study. Then, answer the specific questions below.

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

For the same set of data, wald CI need the least computation time, while Bootstrap t-interval need the most computation time.

- Which method(s) for constructing confidence intervals provide the best coverage when $\epsilon_i \sim N(0, 2)$?

When $\epsilon_i \sim N(0, 2)$, the bootstrap t-interval have the best coverage in most datasets, while bootstrap non-parametric works best when n=100.

- Which method(s) for constructing confidence intervals provide the best coverage when $\epsilon_i \sim logNormal(0, \log(2))$?

When $\epsilon_i \sim logNormal(0, \log(2))$, the wald confidence interval have the best coverage in most datasets, while bootstrap non-parametric works best for others.

In order to save computation time, I set not large enough B and K for bootstrap, which could have affected the performance.