

STAT540 — Computing in Statistics

Bootstrap Confidence Intervals for the Median

Yen-Yi Ho

University of South Carolina

Section 1

Motivation

Why Do We Need Bootstrap?

The median is a great summary — but inference is hard.

For the sample mean $\bar{X}$, the Central Limit Theorem gives us:

$$\bar{X} \sim N\left(\mu, \frac{\sigma^2}{n}\right) \quad \text{for large } n$$

So a 95% CI for μ is simply $\bar{X} \pm 1.96 \cdot s/\sqrt{n}$.

For the sample median $\tilde{X}_n$, the situation is more complicated:

- The sampling distribution depends on the **shape of the density** $f(\cdot)$ at the true median
- The asymptotic SE involves $f(\theta_{0.5})$, which must itself be estimated
- No simple closed-form formula works for all distributions

The Bootstrap Solution

Approximate the sampling distribution of $\tilde{X}_n$ **empirically** by resampling from the data itself.

A Tale of Two Statistics

Sample Mean

- Analytically tractable
- CLT guarantees normality
- $SE = s/\sqrt{n}$
- Standard t -based CI

Works well when data are roughly symmetric.

Sample Median

- Robust to outliers
- Asymptotic $SE \propto 1/f(\theta_{0.5})$
- $f(\cdot)$ must be estimated
- Analytical CI is cumbersome

Works well when data are skewed or heavy-tailed.

Key insight

The median is often the preferred summary for skewed data — which is exactly when analytical inference is hardest. Bootstrap fills that gap.

Section 2

The Bootstrap Idea

The Plug-in Principle

Core idea: use the sample to approximate the population.

	Population	Sample Analogue
Distribution	F (unknown)	$\hat{F}_n$ (empirical CDF)
Parameter	$\theta = T(F)$	$\hat{\theta} = T(\hat{F}_n)$
Sampling dist. of $\hat{\theta}$	unknown	Simulate from $\hat{F}_n$

Sampling from $\hat{F}_n$ is equivalent to **drawing n observations with replacement** from $x_1, x_2, \dots, x_n$.

Efron's Bootstrap (1979)

Approximate the distribution of $\hat{\theta} - \theta$ by the distribution of $\hat{\theta}^* - \hat{\theta}$, where $\hat{\theta}^*$ is computed from a resample drawn from $\hat{F}_n$.

The Bootstrap Algorithm

Given: data $\mathbf{x} = (x_1, \dots, x_n)$ and statistic T (here: median)

- ➊ Compute the **observed statistic**: $\hat{\theta} = \text{median}(\mathbf{x})$
- ➋ **Repeat** B times ($b = 1, 2, \dots, B$):
 - ▶ Draw a **bootstrap sample** $\mathbf{x}^{*b}$ of size n from $\mathbf{x}$ **with replacement**
 - ▶ Compute the **bootstrap replicate**: $\hat{\theta}^{*b} = \text{median}(\mathbf{x}^{*b})$
- ➌ The collection $\{\hat{\theta}^{*1}, \dots, \hat{\theta}^{*B}\}$ approximates the **sampling distribution** of $\hat{\theta}$
- ➍ Extract a **confidence interval** from the bootstrap distribution

Rule of thumb: $B = 1,000$ is often enough; $B = 2,000$ – $5,000$ for stable intervals.

What “With Replacement” Means

Original sample ($n = 8$): 3, 7, 2, 9, 4, 6, 1, 8

Bootstrap sample 1:

3, 3, 7, 2, 4, 4, 8, 6 $\Rightarrow \hat{\theta}^{*1} = 4$

Bootstrap sample 2:

9, 1, 7, 7, 2, 6, 3, 9 $\Rightarrow \hat{\theta}^{*2} = 6.5$

Bootstrap sample 3:

1, 8, 8, 9, 4, 3, 7, 7 $\Rightarrow \hat{\theta}^{*3} = 7$

Key features of each resample:

- Same size n as original
- Some values appear **multiple times**
- Some originals are **absent**
- $\approx 63.2\%$ of original observations appear at least once

$\hat{\theta}^{*1}, \hat{\theta}^{*2}, \dots, \hat{\theta}^{*B}$ form the **bootstrap distribution**.

The Percentile Method for CIs

The simplest way to convert a bootstrap distribution into a CI:

$$95\% \text{ CI} = \left[\hat{\theta}_{(0.025)}^*, \hat{\theta}_{(0.975)}^* \right]$$

where $\hat{\theta}_{(\alpha)}^*$ is the α -quantile of the B bootstrap replicates.

In R:

- `quantile(boot_stats, 0.025)` → lower bound
- `quantile(boot_stats, 0.975)` → upper bound

Intuition: cut the extreme 2.5% from each tail of the bootstrap distribution.

Advantages

- No normality assumption
- Works for skewed distributions
- Easy to compute
- Valid for any statistic

Section 3

R Implementation

Programming Strategy

- ① Set random seed for reproducibility
- ② Record $n = \text{length}(x)$ and the observed median
- ③ **Pre-allocate** a numeric vector of length n_{boot}
- ④ **FOR** b in 1 to n_{boot} :
 - ▶ Draw n indices from $1:n$ **with replacement**
 - ▶ Use those indices to extract a bootstrap sample
 - ▶ Compute the median and store in position b
- ⑤ Compute the 2.5th and 97.5th percentiles of the stored medians
- ⑥ Return: observed median, CI lower, CI upper, full distribution

The Complete Function

```
bootstrap_ci <- function(x, n_boot = 2000, seed = 42) {  
  
  set.seed(seed)  
  n          <- length(x)  
  obs_median <- median(x)  
  boot_meds  <- numeric(n_boot)      # pre-allocate storage  
  
  for (b in 1:n_boot) {  
    idx          <- sample(1:n,      # resample row indices  
                          size       = n,  
                          replace    = TRUE)  
    x_star       <- x[idx]           # bootstrap sample via indexing  
    boot_meds[b] <- median(x_star)   # store one bootstrap replicate  
  }  
  
  ci <- quantile(boot_meds, probs = c(0.025, 0.975))  
  
  list(  
    estimate = obs_median,  
    ci_lower = ci[[1]],  
    ci_upper = ci[[2]],  
    boot_dist = boot_meds      # keep for diagnostic plots  
  )  
}
```

Line-by-Line: Setup and Pre-allocation

```
bootstrap_ci <- function(x, n_boot = 2000, seed = 42) {  
  set.seed(seed)  
  n          <- length(x)  
  obs_median <- median(x)  
  boot_meds  <- numeric(n_boot)
```

- `x` — the numeric data vector (any length)
- `n_boot = 2000` — default resamples; caller can override
- `seed = 42` — placed *inside* the function so the same seed applies every time the function is called, regardless of the caller's RNG state
- `numeric(n_boot)` — creates a vector of B zeros before the loop; filling by index is $O(1)$ per iteration instead of $O(B)$
- We compute `obs_median` once here rather than inside the loop

Line-by-Line: The Loop and CI

```
for (b in 1:n_boot) {  
  idx      <- sample(1:n, size = n, replace = TRUE)  
  x_star   <- x[idx]  
  boot_meds[b] <- median(x_star)  
}  
  
ci <- quantile(boot_meds, probs = c(0.025, 0.975))  
  
list(estimate = obs_median,  
     ci_lower  = ci[[1]],  
     ci_upper  = ci[[2]],  
     boot_dist = boot_meds)
```

- `sample(1:n, size = n, replace = TRUE)` — draws n integer indices; sampling indices (not values directly) generalises to data frames
- `x[idx]` — indexing into `x` with those integers gives the bootstrap sample
- `boot_meds[b] <- ...` — assignment into a pre-allocated slot; the vector never grows
- `quantile(..., 0.025, 0.975)` — the percentile CI in one call
- Returning `boot_dist` lets the caller plot the bootstrap distribution for diagnostics

Common Mistakes to Avoid

Not efficient — growing the vector:

```
boot_meds <- c()
for (b in 1:n_boot) {
  ...
  boot_meds <- c(boot_meds,
                 median(x_star))
}
```

Copies the entire vector on each of B iterations.
Very slow for large B .

Better — pre-allocate:

```
boot_meds <- numeric(n_boot)
for (b in 1:n_boot) {
  ...
  boot_meds[b] <- median(x_star)
}
```

Assigns in place. Constant memory per iteration.

Two more pitfalls

- Forgetting `replace = TRUE` gives a permutation, *not* a bootstrap resample — every sample will have the same median as the original data.
- Placing `set.seed()` *outside* the function means results change depending on the caller's RNG state.

Section 4

Applied Example

The Data: NHANES Adult BMI

The **NHANES** package ships a 10,000-row extract of the US National Health and Nutrition Examination Survey. We analyse **BMI (kg/m²)** among adults.

n = 4786

Mean = 28.79

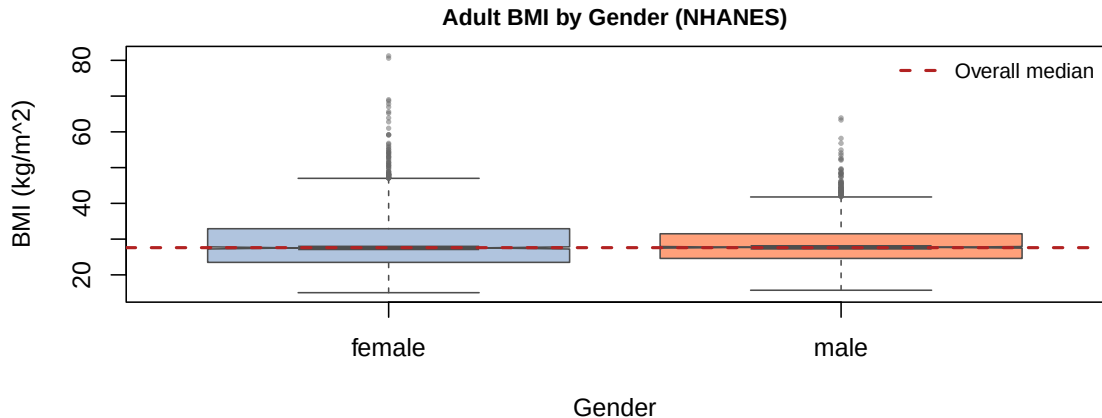
Median = 27.61

SD = 6.84

Skewness = 1.372

BMI is right-skewed — a long upper tail of high values pulls the mean above the median, so the median is the more representative centre and bootstrap inference is appropriate here.

BMI by Gender



Both groups carry the same right skew: whiskers stretch further upward and the outliers sit almost entirely above the box.

Running `bootstrap_ci()`

```
result <- bootstrap_ci(bmi, n_boot = 2000, seed = 42)

cat("Observed median :", round(result$estimate, 2), "\n")
```

Observed median : 27.61

```
cat("95% Bootstrap CI: [",
    round(result$ci_lower, 2), ", ",
    round(result$ci_upper, 2), "]\n")
```

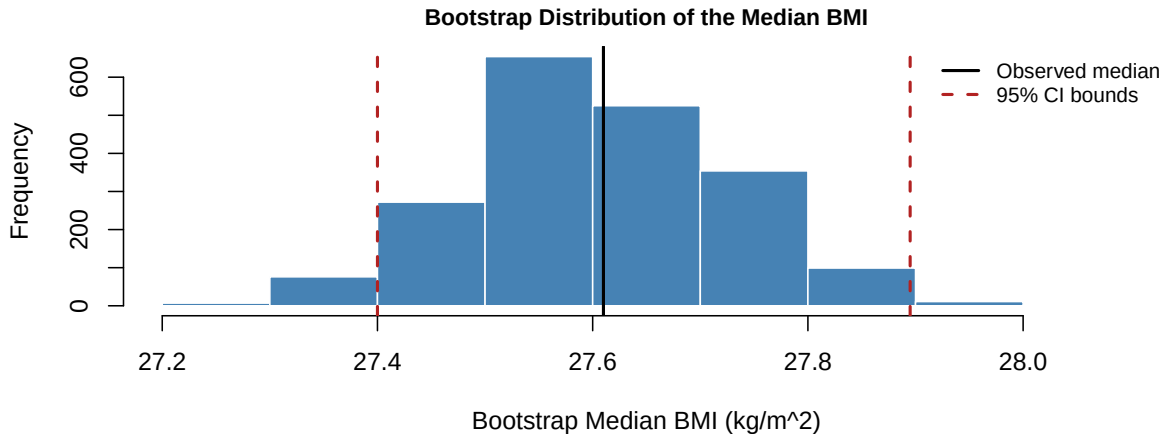
95% Bootstrap CI: [27.4 , 27.9]

Interpretation: We are 95% confident that the median BMI among US adults lies between 27.4 and 27.9 kg/m².

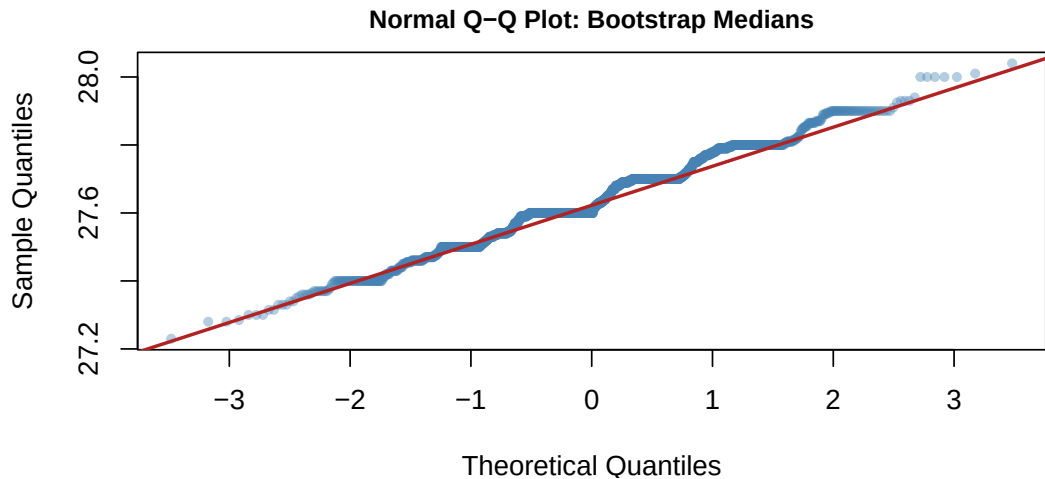
Interval width

Width = 0.5 kg/m². With n in the thousands the interval is narrow — precision of the median improves at rate $1/\sqrt{n}$, just as it does for the mean.

Visualising the Bootstrap Distribution



Checking Normality of the Bootstrap Distribution



If approximately linear: the normal-theory CI would give similar results. BMI is recorded to one decimal place, so the bootstrap medians are **discrete** — the Q-Q plot shows visible steps.

Sensitivity to Number of Resamples B

Table 1: 95% Bootstrap CI by number of resamples

B	Lower	Upper	Width
200	27.4	27.85	0.45
500	27.4	27.90	0.50
1000	27.4	27.90	0.50
2000	27.4	27.90	0.50
5000	27.4	27.90	0.50

The interval stabilises quickly. $B = 2,000$ is sufficient here.

Section 5

Key Takeaways

Summary

Conceptual

- Bootstrap replaces unknown F with $\hat{F}_n$
- Resampling *with replacement* simulates repeated sampling
- Percentile method: cut 2.5% from each tail
- No distributional assumptions required
- Especially useful when the statistic's SE has no clean formula

Programming

- Wrap logic in a **function** for reusability
- **Pre-allocate** storage before the loop
- Sample **indices**, then index: `x[idx]`
- Set seed *inside* the function
- Return `boot_dist` for diagnostics
- $B = 2,000$ is a safe default

When to use bootstrap CIs for the median

Data are skewed or heavy-tailed; normality of the statistic's sampling distribution is in doubt; the analytical SE requires estimating $f(\theta_{0.5})$; or sample size is moderate and you want to avoid parametric assumptions.

Session Info

R version 4.4.0 (2024-04-24)

Platform: aarch64-apple-darwin20

Running under: macOS 26.6.2

Matrix products: default

BLAS: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libR

LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libR

locale:

[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/New_York

tzcode source: internal

attached base packages:

[1] stats graphics grDevices utils datasets methods base