Bootstrap-t Comparison

Mario Ibanez

March 26, 2016

Question

Let $x_1, x_2, ...x_{20}$ each be independent values from an exponential distribution with mean $= 1/lambda$. (In other words, $f(x) = \lambda e^{-\lambda x}$). Use the Bootstrap-t method to construct a 95% confidence interval for the parameter λ.

Answer

Below is a random sample of 20 values from the exponential distribution:

```
# Set seed
set.seed(1234)

# Load packages and data
library(ggplot2)
library(knitr)

# 20 random exponential values with mean = 1/10
lambda <- 10
data <- rexp(n = 20, rate = lambda); data
```

```
##  [1] 0.2501758605 0.0246758883 0.0006581957 0.1742746090 0.0387182584
##  [6] 0.0089949671 0.0824081515 0.0202617901 0.0838040319 0.0760430301
## [11] 0.1880076678 0.1596105418 0.1658662384 0.3052458100 0.1750680133
## [16] 0.0031725529 0.0876960574 0.0014613740 0.1835064027 0.0519341271
```

The 95% confidence interval will be calcuated two ways.

Method 1

This method will use the delta method to estimate the variance of $\hat{\theta} = 1/\overline{x}$. We have:

$$S_{\hat{\theta}} = \frac{S}{\overline{x}^2\sqrt{n}}$$

$$\hat{\theta} = \frac{1}{\overline{x}}$$

Below are the values of $\hat{\theta}$ and $S_{\hat{\theta}}$

```
# Values for original sample
theta_hat <- 1 / mean(data); theta_hat
```

```
## [1] 9.60807
```

```
s_theta_hat <- sd(data) / (mean(data)^2 * sqrt(length(data))); s_theta_hat
```

```
## [1] 1.862861
```

Now we can bootstrap 1000 samples from this data. First, this is the function that will be used to generate the "t-statistic" for each bootstrapped sample:

```
# Function to calculate statistics
statistic_generator <- function(x, theta_hat){
  theta_hat_star <- 1 / mean(x)
  s_theta_hat_star <- sd(x) / (mean(x)^2 * sqrt(length(x)))
  return((theta_hat_star - theta_hat)/s_theta_hat_star)
}
```

This function will calculate the values $th\hat{e}ta$, $S_{\hat{\theta}}$, and T_i^* when given a bootstrapped sample.

Now 1000 bootstrapped samples are calculated:

```
boot_strapped_data <- matrix(data = sample(x = data, size = 20*1000, replace = TRUE),
                             ncol = 20, byrow = TRUE)
```

Now the statistics are calculated:

```
boot_statistics <- matrix(apply(X = boot_strapped_data, MARGIN = 1,
                                FUN = statistic_generator, theta_hat = theta_hat),
                          ncol = 1, byrow = TRUE)
```

Now we can calcuate the 95% confidence interval using the 2.5th and 97.5th percentiles of the "t-statistics":

```
sorted_t_statistics <- sort(boot_statistics[, 1])
lower_limit <- theta_hat - sorted_t_statistics[1000*0.975] * s_theta_hat; lower_limit
```

```
## [1] 6.98008
```

```
upper_limit <- theta_hat - sorted_t_statistics[1000*0.025] * s_theta_hat; upper_limit
```

```
## [1] 15.10695
```

Now we can try to run this same method 100 times in order to approximate the true coverage probability of this method.

```
results_vector <- rep(0, 100)
confidence_width <- rep(0, 100)

for(i in 1:100){
  boot_strapped_data <- matrix(data = sample(x = data, size = 20*1000, replace = TRUE),
                               ncol = 20, byrow = TRUE)
```

```r
  boot_statistics <- matrix(apply(X = boot_strapped_data, MARGIN = 1,
                                  FUN = statistic_generator, theta_hat = theta_hat),
                              ncol = 1, byrow = TRUE)

  sorted_t_statistics <- sort(boot_statistics[, 1])
  lower_limit <- theta_hat - sorted_t_statistics[1000*0.975] * s_theta_hat; lower_limit
  upper_limit <- theta_hat - sorted_t_statistics[1000*0.025] * s_theta_hat; upper_limit

  results_vector[i] <- ((lambda < upper_limit) && (lambda > lower_limit))
  confidence_width[i] <- upper_limit - lower_limit
}
```

Out of 100 simulations, 100 contained the true value of lambda. The average confidence interval with was 7.7427197

Method 2

This method creates a confidence interval instead for the value $1/\lambda$ directly, in otherwords, $\hat{\theta} = \overline{x}$. At the end, the confidence interval is transformed into a confidence interval for λ

```r
# Values for original sample
theta_hat2 <- mean(data); theta_hat2
```

```
## [1] 0.1040792
```

```r
s_theta_hat2 <- sd(data) / sqrt(length(data)); s_theta_hat2
```

```
## [1] 0.0201794
```

Now we can bootstrap 1000 samples from this data. First, this is the function that will be used to generate the "t-statistic" for each bootstrapped sample:

```r
# Function to calculate statistics
statistic_generator2 <- function(x, theta_hat2){
  theta_hat_star <- mean(x)
  s_theta_hat_star <- sd(x) / sqrt(length(x))
  return((theta_hat_star - theta_hat2)/s_theta_hat_star)
}
```

So that the comparison is more equivalent, the same bootstrapped data will be used. Or course new "t-statistics" need to be calculated however.

```r
boot_statistics2 <- matrix(apply(X = boot_strapped_data, MARGIN = 1,
                                 FUN = statistic_generator2, theta_hat = theta_hat2),
                          ncol = 1, byrow = TRUE)
```

Now we can calcuate the 95% confidence interval using the 2.5th and 97.5th percentiles of the "t-statistics":

```
sorted_t_statistics2 <- sort(boot_statistics2[, 1])
lower_limit2 <- theta_hat2 - sorted_t_statistics2[1000*0.975] * s_theta_hat2; lower_limit2
```

```
## [1] 0.0642807
```

```
upper_limit2 <- theta_hat2 - sorted_t_statistics2[1000*0.025] * s_theta_hat2; upper_limit2
```

```
## [1] 0.1521753
```

These lower and upper limits are for the value of $1/\lambda$.

Now we can try to run this same method 100 times in order to approximate the true coverage probability of this method.

```
results_vector2 <- rep(0, 100)
confidence_width2 <- rep(0, 100)

for(i in 1:100){
  boot_strapped_data <- matrix(data = sample(x = data, size = 20*1000, replace = TRUE),
                               ncol = 20, byrow = TRUE)

  boot_statistics <- matrix(apply(X = boot_strapped_data, MARGIN = 1,
                                FUN = statistic_generator2, theta_hat = theta_hat2),
                            ncol = 1, byrow = TRUE)

  sorted_t_statistics <- sort(boot_statistics[, 1])
  upper_limit <- 1 / (theta_hat2 - sorted_t_statistics[1000*0.975] * s_theta_hat2)
  lower_limit <- 1 / (theta_hat2 - sorted_t_statistics[1000*0.025] * s_theta_hat2)

  results_vector2[i] <- ((lambda < upper_limit) && (lambda > lower_limit))
  confidence_width2[i] <- upper_limit - lower_limit
}
```

Out of 100 simulations, 100 contained the true value of lambda. The average confidence interval with was 8.6291913

This method also had good performance but the confidence intervals were much wider on average.