Statistical Methods for Data Science
CS 6313.001: Mini Project #6

Due on Thursday May 4, 2017 at 4pm

Instructor: Pankaj Choudhary

Hanlin He / Lizhong Zhang (hxh160630 / lxz160730)

Contents

Contribution

Both team members made the same contribution in this project.

1 Answers

- Bias is -0.003031904 and standard error 0.09476028.
- 2.5th and 97.5th percentile of $\hat{\theta}$: $[3.669463, 4.065631]$
- 2.5th and 97.5th percentile of $\hat{\theta} - \theta$: $[-0.1871917, 0.1872621]$
- CI:
  - Normal Approximation CI: $[3.68736, 4.066203]$
  - Basic Bootstrap CI: $[3.68736, 4.066203]$
  - Normal Approximation CI: $[3.68736, 4.066203]$

The histogram and qqplot for bootstrap distribution is shown in fig. 1.

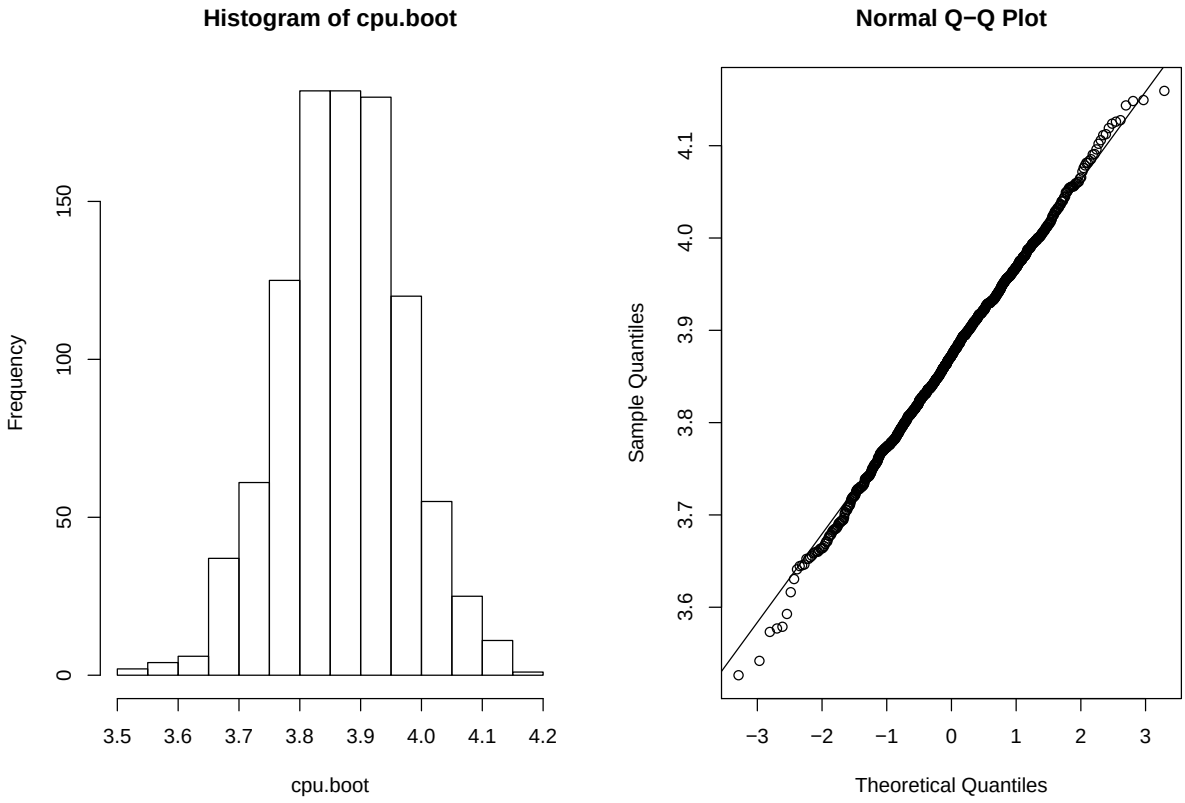


Figure 1: Histogram & QQplot

The bootstrap CI and plot indicates that $\log(mean(\hat{\theta}))$ is a good estimator of the parameter of interest.

2 R Code

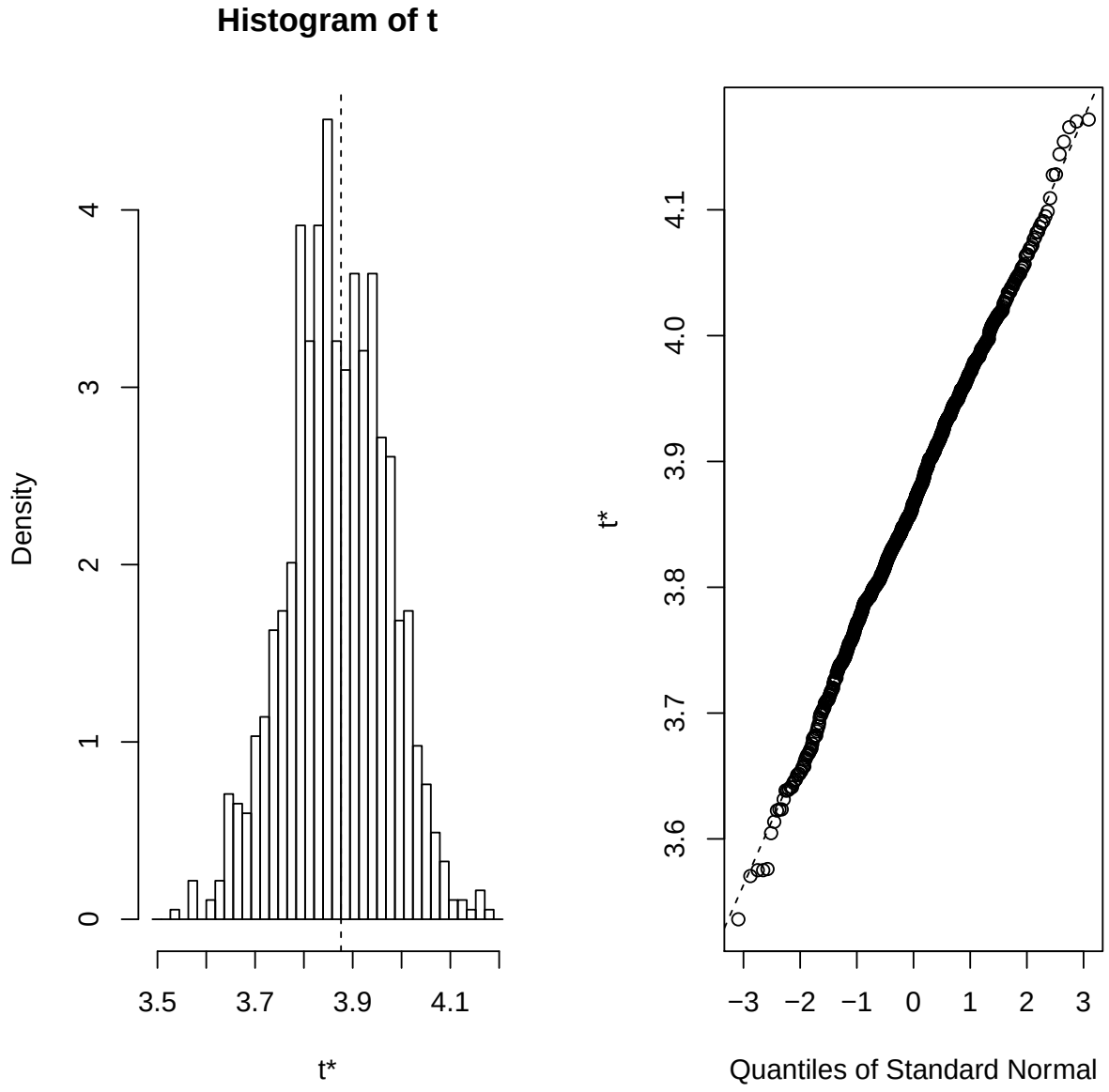
```r
cpu <- scan(file="cputime.txt")

# Define the function to compute the natural logarithm of the polulation mean.
logmean <- function(data) {
  return(log(mean(data)))
}

# Define the resample function.
resample <- function(x) {
  sample(x, size = length(x), replace = TRUE)
}

# Define the bootstrap function.
rboot <- function(B, statistic, simulator, data) {
  tboots <- replicate(B, statistic(simulator(data)))
  return(tboots)
}

# Define the bootstrap function that directly compute specific statistic.
bootstrap.se <- function(B, statistic, simulator, data) {
  tboots <- rboot(B, statistic, simulator, data)
  se <- sd(tboots)
  return(se)
}

# Define the bootstrap function that directly compute the bias of specific statistic.
bootstrap.bias <- function(B, statistic, simulator, t.hat, data) {
  tboots <- rboot(B, statistic, simulator, data)
  bias <- mean(tboots) - t.hat
  return(bias)
}

# Define the bootstrap function that directly compute the quantile
# of specific statistic.
bootstrap.alphaquantile <- function(B, statistic, simulator, data, alpha) {
  tboots <- rboot(B, statistic, simulator, data)
  alphaquantile <- quantile(tboots, alpha)
  return(alphaquantile)
}

# Define the bootstrap function that directly compute the quantile
```

```r
# of the bias of specific statistic.
bootstrap.alphaquantile.hat <- function(B, statistic, simulator, t.hat, data, alpha) {
  alphaquantile <- bootstrap.alphaquantile(B, statistic, simulator, data, alpha)
  alphaquantile.hat <- alphaquantile - t.hat
  return(alphaquantile.hat)
}

# Define function to compute the CI with normal approximation.
NormalCI <- function(B, statistic, simulator, t.hat, data, alpha) {
  tboots <- rboot(B, statistic, simulator, data)
  se <- sd(tboots)
  bias <- mean(tboots) - t.hat
  ci.lower <- t.hat - bias - qnorm(1 - alpha/2) * se
  ci.upper <- t.hat - bias - qnorm(alpha/2) * se
  return(list(ci.lower=ci.lower,ci.upper=ci.upper))
}

# Define function to compute the CI with basic bootstrap.
BasicBootstrapCI <- function(B, statistic, simulator, t.hat, data, alpha) {
  tboots <- rboot(B, statistic, simulator, data)
  ci.lower <- 2 * t.hat - quantile(tboots, 1 - alpha/2)
  ci.upper <- 2 * t.hat - quantile(tboots, alpha/2)
  return(list(ci.lower=ci.lower,ci.upper=ci.upper))
}

# Define function to compute the CI with percentile bootstrap.
PercentileBootstrapCI <- function(B, statistic, simulator, data, alpha) {
  tboots <- rboot(B, statistic, simulator, data)
  ci.lower <- quantile(tboots, alpha/2)
  ci.upper <- quantile(tboots, 1 - alpha/2)
  return(list(ci.lower=ci.lower,ci.upper=ci.upper))
}

# Compute t.hat
t.hat <- logmean(cpu)

# Compute bootstrap distribution.
cpu.boot <- rboot(1000, logmean, resample, cpu)

# Compute standard error of theta hat.
t.hat.se <- bootstrap.se(1000, logmean, resample, cpu)

# Compute bias of theta hat.
```

```r
t.hat.bias <- bootstrap.bias(1000, logmean, resample, logmean(cpu), cpu)

# Compute 2.5th and 97.5th percentiles of the theta hat.
bootstrap.alphaquantile(1000, logmean, resample, cpu, 0.025)

##     2.5%
## 3.692207

bootstrap.alphaquantile(1000, logmean, resample, cpu, 0.975)

##    97.5%
## 4.050683

# Compute 2.5th and 97.5th percentiles of the bias of theta hat.
bootstrap.alphaquantile.hat(1000, logmean, resample, logmean(cpu), cpu, 0.025)

##       2.5%
## -0.1946992

bootstrap.alphaquantile.hat(1000, logmean, resample, logmean(cpu), cpu, 0.975)

##     97.5%
## 0.1838178

# Compute normal approximation of CI.
NormalCI(1000, logmean, resample, logmean(cpu), cpu, 0.05)

## $ci.lower
## [1] 3.692305
##
## $ci.upper
## [1] 4.06934

# Compute basic bootstrap CI.
BasicBootstrapCI(1000, logmean, resample, logmean(cpu), cpu, 0.05)

## $ci.lower
##    97.5%
## 3.689361
##
## $ci.upper
##     2.5%
## 4.067459

# Compute normal approximation of CI.
PercentileBootstrapCI(1000, logmean, resample, cpu, 0.05)
```

```
## $ci.lower
##     2.5%
## 3.668655
##
## $ci.upper
##    97.5%
## 4.044892

# Draw histogram and qqplot for bootstrap distribution
# Create fig folder to store plot.
if(!dir.exists("fig")) dir.create("fig")
pdf("fig/hist.pdf", width=5, height=7)
hist(cpu.boot)
dev.off()

## pdf
##   2

pdf("fig/qqplot.pdf", width=5, height=7)
qqnorm(cpu.boot)
qqline(cpu.boot)
dev.off()

## pdf
##   2

# Verification
library(boot)
nlm.hat <- function(x, indices){
  result <- log(mean(x[indices]))
  return(result)
}
nlm.hat.boot <- boot(cpu, nlm.hat, R = 1000, sim = "ordinary", stype = "i")
names(nlm.hat.boot)

##  [1] "t0"        "t"         "R"         "data"      "seed"
##  [6] "statistic" "sim"       "call"      "stype"     "strata"
## [11] "weights"

log(mean(cpu))

## [1] 3.87605

nlm.hat.boot$t0

## [1] 3.87605
```

```r
mean(nlm.hat.boot$t) - nlm.hat.boot$t0

## [1] -0.007910493

quantile(mean(nlm.hat.boot$t) - nlm.hat.boot$t0,0.975)

##        97.5%
## -0.007910493

sd(nlm.hat.boot$t)

## [1] 0.1015838

plot(nlm.hat.boot)
```




```
boot.ci(nlm.hat.boot)
```

```
## Warning in boot.ci(nlm.hat.boot): bootstrap variances needed for studentized intervals
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = nlm.hat.boot)
##
## Intervals :
## Level      Normal              Basic
```

```
## 95%   ( 3.685,  4.083 )   ( 3.695,  4.095 )
##
## Level     Percentile            BCa
## 95%   ( 3.657,  4.057 )   ( 3.690,  4.091 )
## Calculations and Intervals on Original Scale
```