The bootstrap

Here is a series of exercises on the non-parametric bootstrap. We will first empirically derive the probability that a given observation is part of a bootstrap sample.

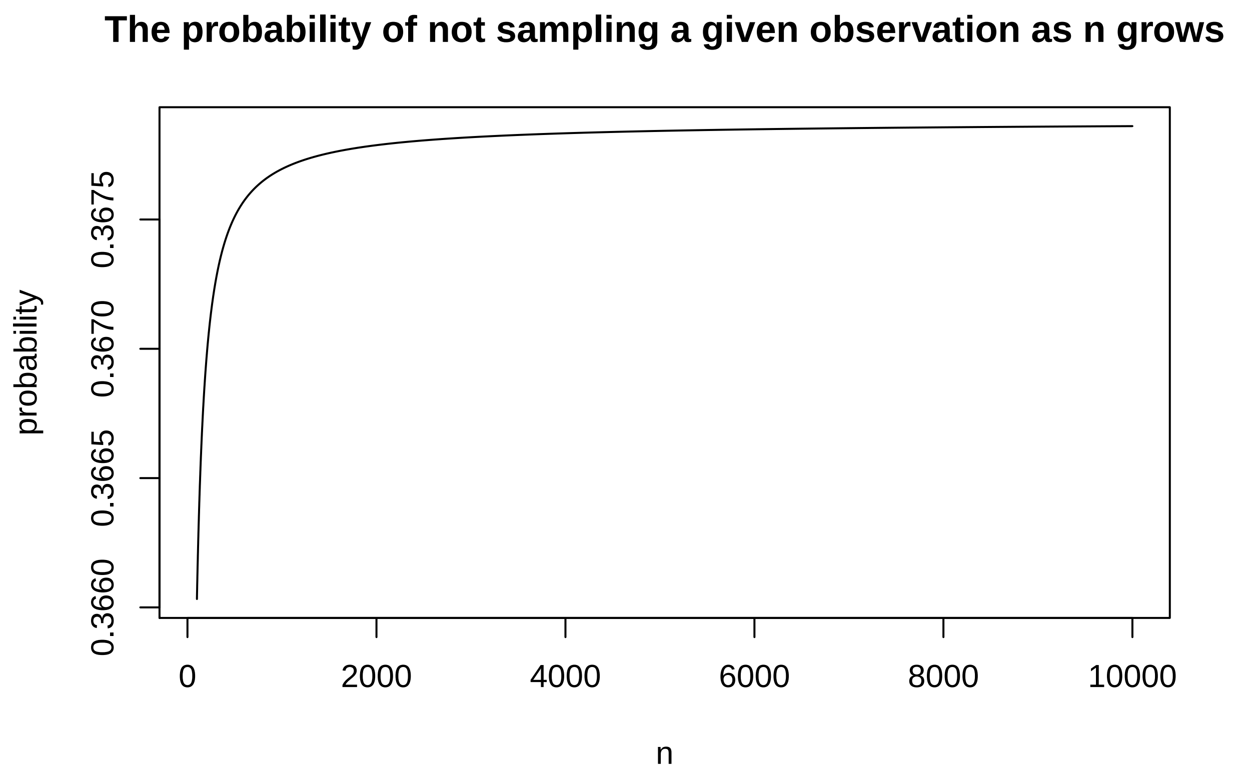
```r
# sampling with replacement
p.not.in.sample <- function(n){
  return((1-(1/n))**n)
}
```

Example: what is the probability that a given observation is not in the sample, if it has size 100?

```r
p.not.in.sample(100)
```

```
## [1] 0.3660323
```

```r
# now let's simulate and plot
plot(100:10000, p.not.in.sample(100:10000), main="The probability of not sampling a given observation a
```



Looking at the above result we can answer the following question: what proportion of the original observations do you expect to be in a bootstrap sample of size n?

On average, weighting each observation as 1/n, we would have that the number of observations left out would be approximately equal to 1/3 (take the expectation of the indicator variable to see it). Hence,on average, we would expect to have 2/3 of the original dataset in a bootstrap sample.

Empirical coverage of Bootstrap confidence intervals

We want to estimate the trimmed mean of the Gamma distribution where the 10% largest and 10% smallest observations are trimmed.

```r
set.seed(0)
# approximate true parameter value with a huge sample
true.tm <-mean(rgamma(100000000, shape = 2, rate = 1), trim = 0.1)
true.tm
```

```
## [1] 1.820736
```

So true.tm is our ground truth. We're now going to draw a small sample from the true distribution to use it to do inference.

```r
n<-40
small.sample <- rgamma(n,shape=2,rate=1)
# our sample estimate
hat.tm <- mean(small.sample, trim = 0.1)
hat.tm
```

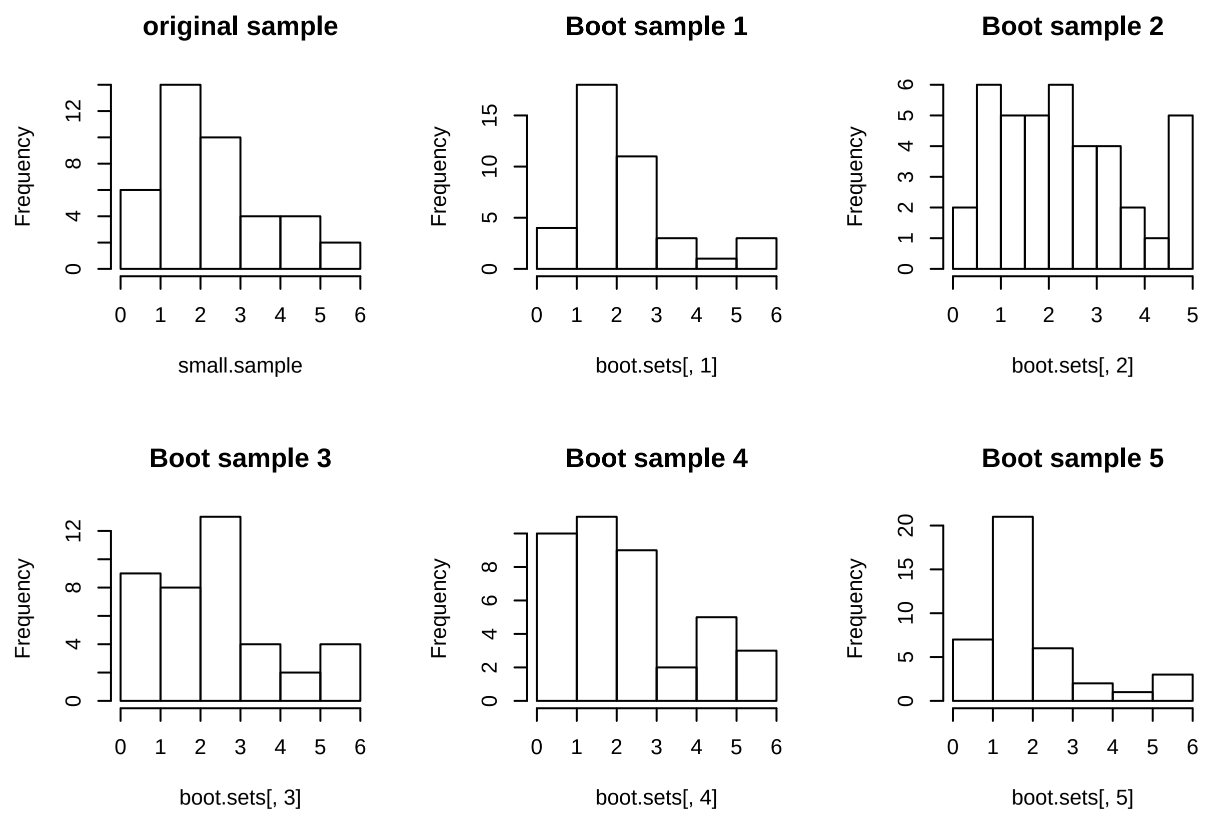
```
## [1] 2.171702
```

We'll now use Bootstrap to build 95% confidence intervals around our estimate.

```r
# First of all, we need to create B bootstrap sets.
B <- 50
boot.sets <- matrix(nrow=n,ncol=B)
for(b in 1:B){
  b.set <- sample(small.sample,size=n, replace = T)
  boot.sets[,b]<- b.set
}
```

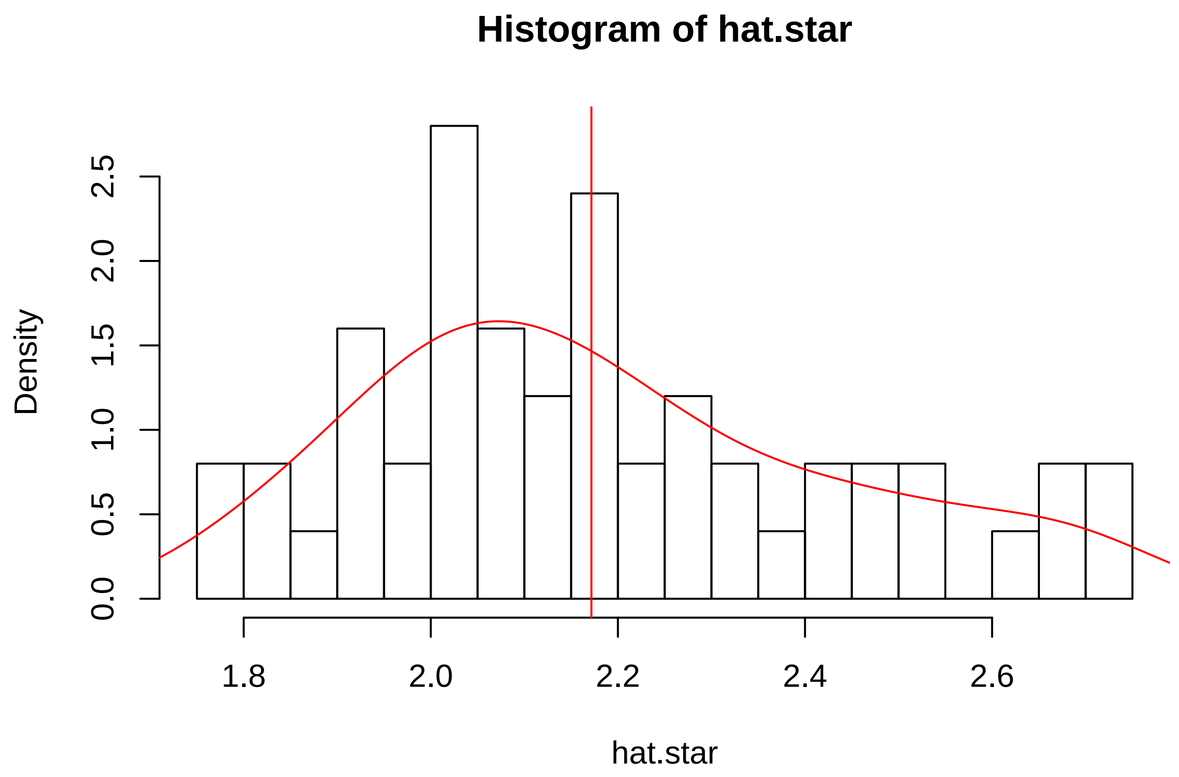
Let's visualize the first five datasets

```r
par(mfrow=c(2,3))
hist(small.sample, main="original sample")
hist(boot.sets[,1], main="Boot sample 1")
hist(boot.sets[,2], main="Boot sample 2")
hist(boot.sets[,3], main="Boot sample 3")
hist(boot.sets[,4], main="Boot sample 4")
hist(boot.sets[,5], main="Boot sample 5")
```



Okay now we're ready to create the confidence intervals. We're going to create four different CIs.

```r
hat.star <- matrix(nrow = B, ncol=1)
for(b in 1:B){
  hat.star[b]<-mean(boot.sets[,b],trim=0.1)
}
hist(hat.star, probability = T, breaks=20)
abline(v=hat.tm, col="red")
lines(density(hat.star), col="red")
```

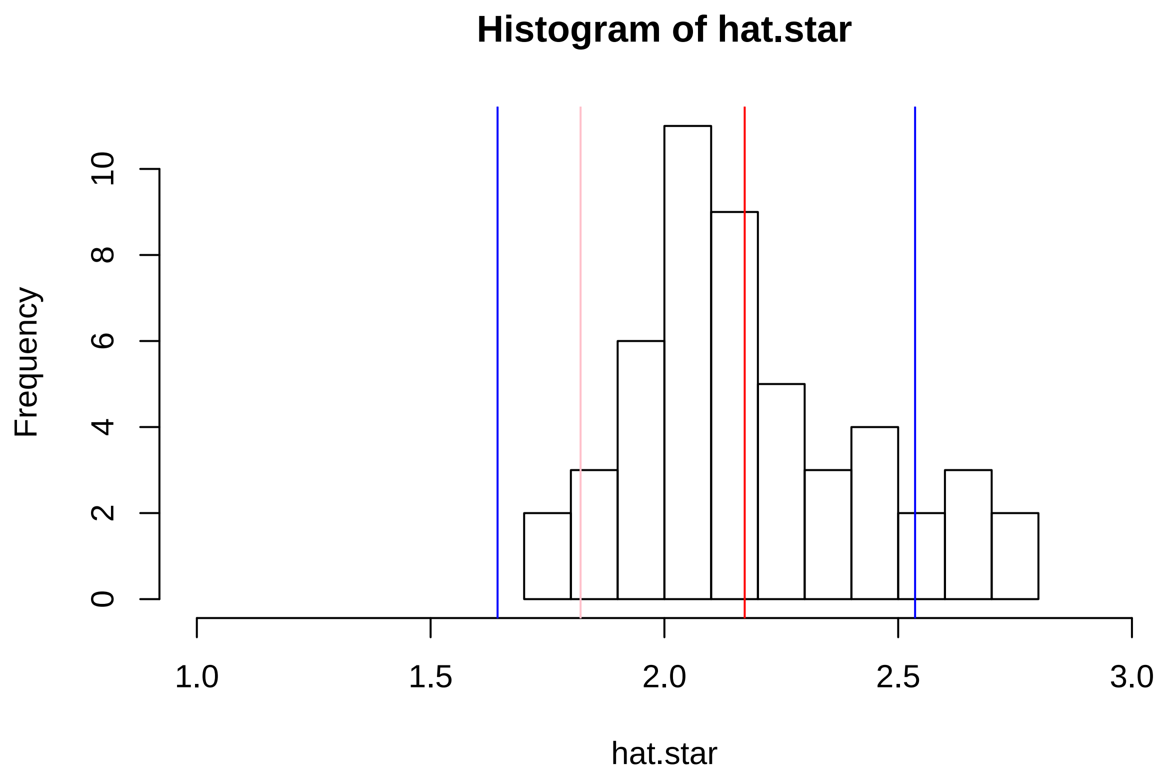


The reversed quantile CI

```
# find the empirical quantiles of the bootstrap distribution
p.star <- hat.star - hat.tm
lower.q <- quantile(p.star, probs = 0.025)
upper.q <- quantile(p.star, probs = 0.975)
lower <- hat.tm - upper.q
upper <- hat.tm - lower.q
c(lower,upper)
```

```
##    97.5%     2.5%
## 1.643181 2.536170
```

```
hist(hat.star, xlim = c(1,3))
abline(v=lower, col="blue")
abline(v=upper, col="blue")
abline(v=hat.tm, col="red")
abline(v=true.tm, col="pink")
```



Let's check our results against R results.

```
require("boot")
```

```
## Loading required package: boot
```

```
# statistic functions
tm.fun <- function(x, ind){return(mean(x[ind], trim=0.1))}
tm.var <- function(x, ind){
  tm.value <- tm.fun(x[ind])
  # second level bootstrap to obtain variance of our estimate
  tm.variance <- var(boot(data=x[ind],R=50,statistic=tm.fun)$t)
  return(c(tm.value,tm.variance))
}
```

```
boot.res <- boot(data=small.sample,statistic=tm.fun, R=50, sim="ordinary")
boot.ci(boot.res, conf=0.95, type="perc")
```

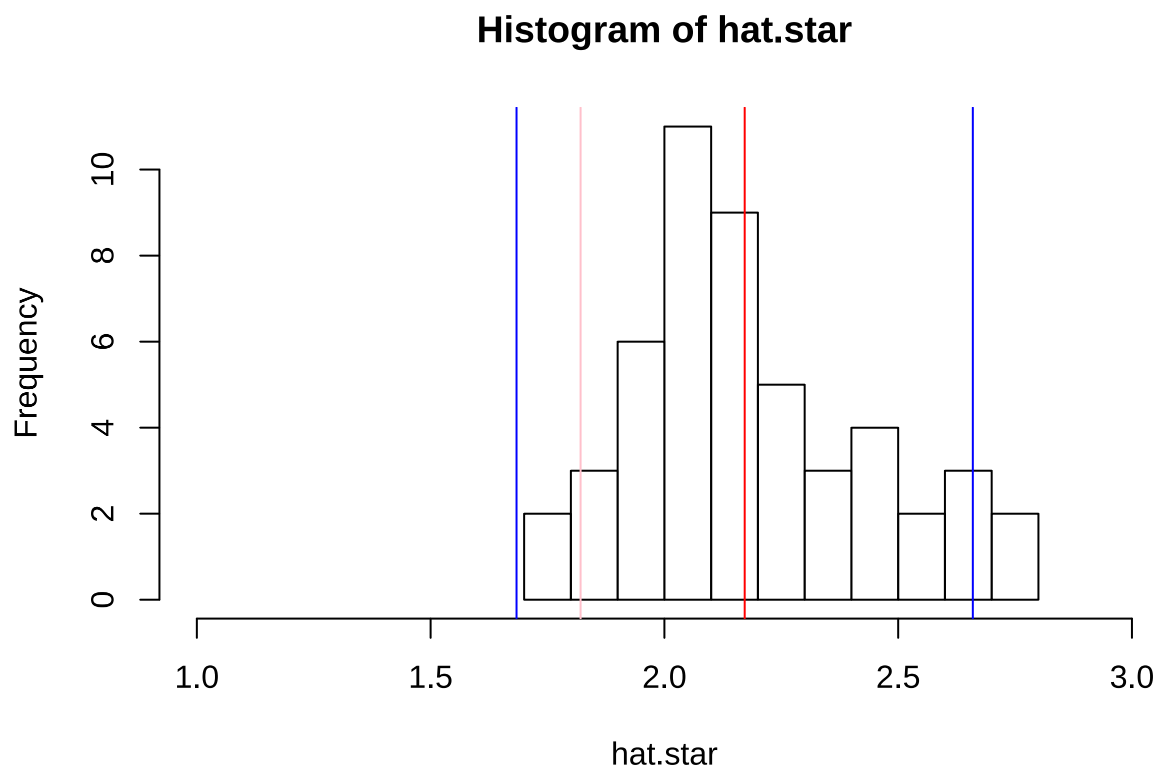
```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 50 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.res, conf = 0.95, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   ( 1.818,  2.714 )
## Calculations and Intervals on Original Scale
## Some percentile intervals may be unstable
```

The normal approximation CI

```
# this one is base on the assumption that the estimate distribution tends to a Gaussian as n grows
sd.hat.tm <- sqrt(var(hat.star))
lower.q <- qnorm(0.025)*sd.hat.tm
lower <- hat.tm + lower.q
upper <- hat.tm - lower.q
c(lower,upper)
```

```
## [1] 1.683751 2.659654
```

```
hist(hat.star, xlim = c(1,3))
abline(v=lower, col="blue")
abline(v=upper, col="blue")
abline(v=hat.tm, col="red")
abline(v=true.tm, col="pink")
```



Now let's check with the R results:

```
boot.ci(boot.res, conf=0.95, type="norm")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 50 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.res, conf = 0.95, type = "norm")
```

```
##
## Intervals :
## Level      Normal
## 95%   ( 1.742,  2.534 )
## Calculations and Intervals on Original Scale
```

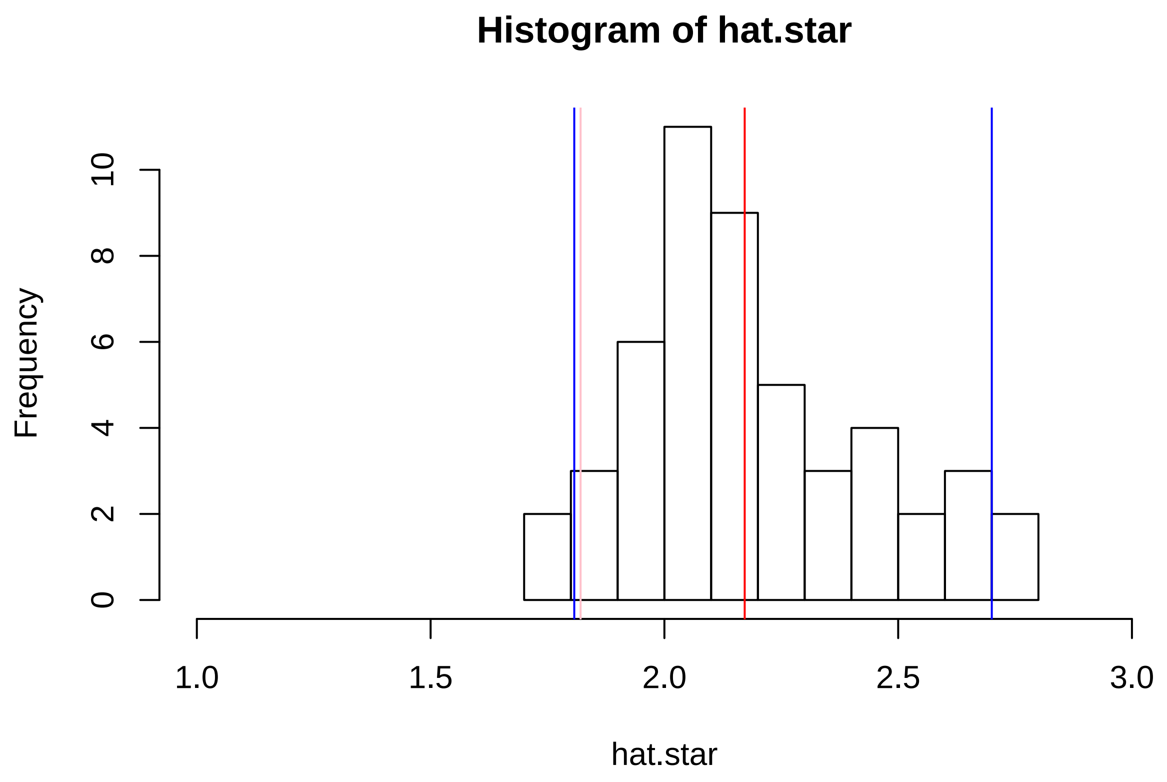
The naive CI

The naive CI directly uses the hat.star quantiles. Note: no theoretical justification unless it's distribution is symmetric.

```
lower.q <- quantile(hat.star, probs = 0.025)
upper.q <- quantile(hat.star, probs = 0.975)
lower <- lower.q
upper <- upper.q
c(lower,upper)
```

```
##     2.5%    97.5%
## 1.807235 2.700224
```

```
hist(hat.star, xlim = c(1,3))
abline(v=lower, col="blue")
abline(v=upper, col="blue")
abline(v=hat.tm, col="red")
abline(v=true.tm, col="pink")
```



Again let's check with the R results:

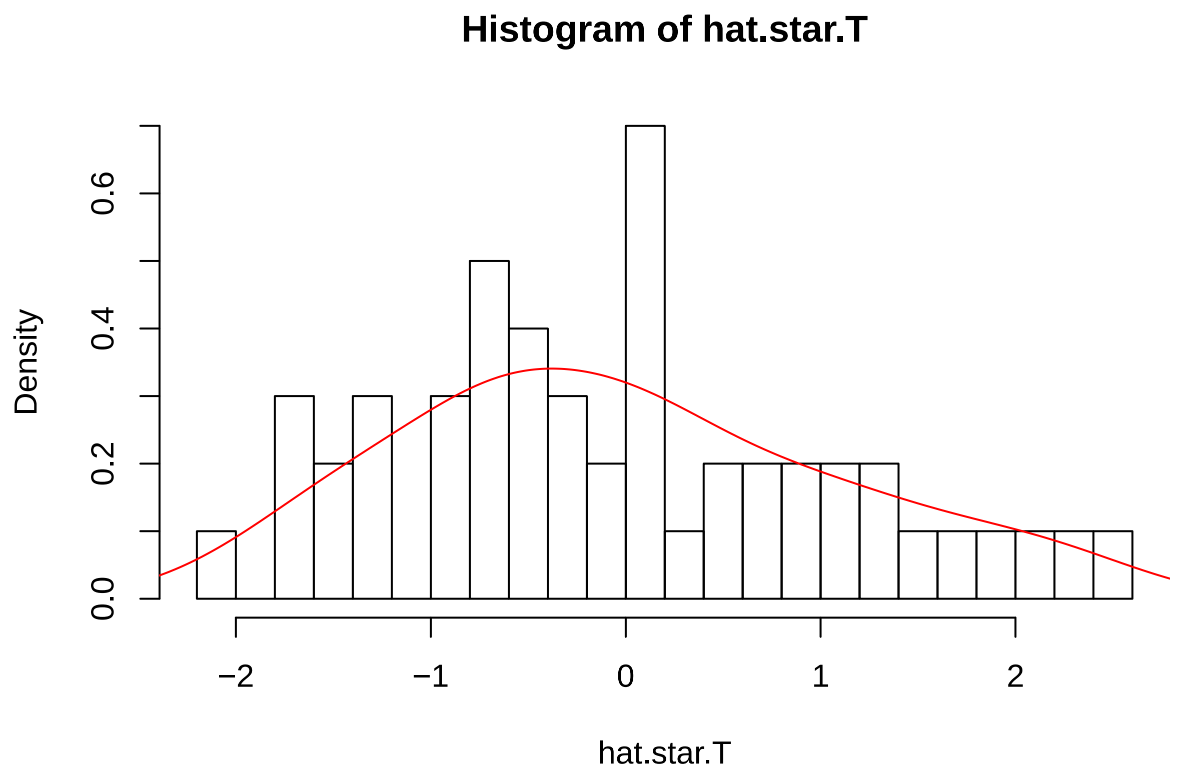
```r
boot.ci(boot.res, conf=0.95, type="basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 50 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.res, conf = 0.95, type = "basic")
##
## Intervals :
## Level      Basic
## 95%   ( 1.629,  2.525 )
## Calculations and Intervals on Original Scale
## Some basic intervals may be unstable
```

The bootstrap T CI

Now, to obtain the bootstrap T CI we need a second level bootstrap.

```r
C <- 50
hat.star.T <- matrix(nrow=B, ncol=1)
for(b in 1:B){
  hat.star.b<-matrix(nrow=C,ncol=1)
  for(c in 1:C){
    c.set <- sample(boot.sets[,b],size=n, replace = T)
    hat.star.b[c]<-mean(c.set,trim=0.1)
  }
  sd.b<-sqrt(var(hat.star.b))
  hat.star.T[b]<-(hat.star[b]-hat.tm)/sd.b
}
hist(hat.star.T, probability = T, breaks=20)
lines(density(hat.star.T), col="red")
```

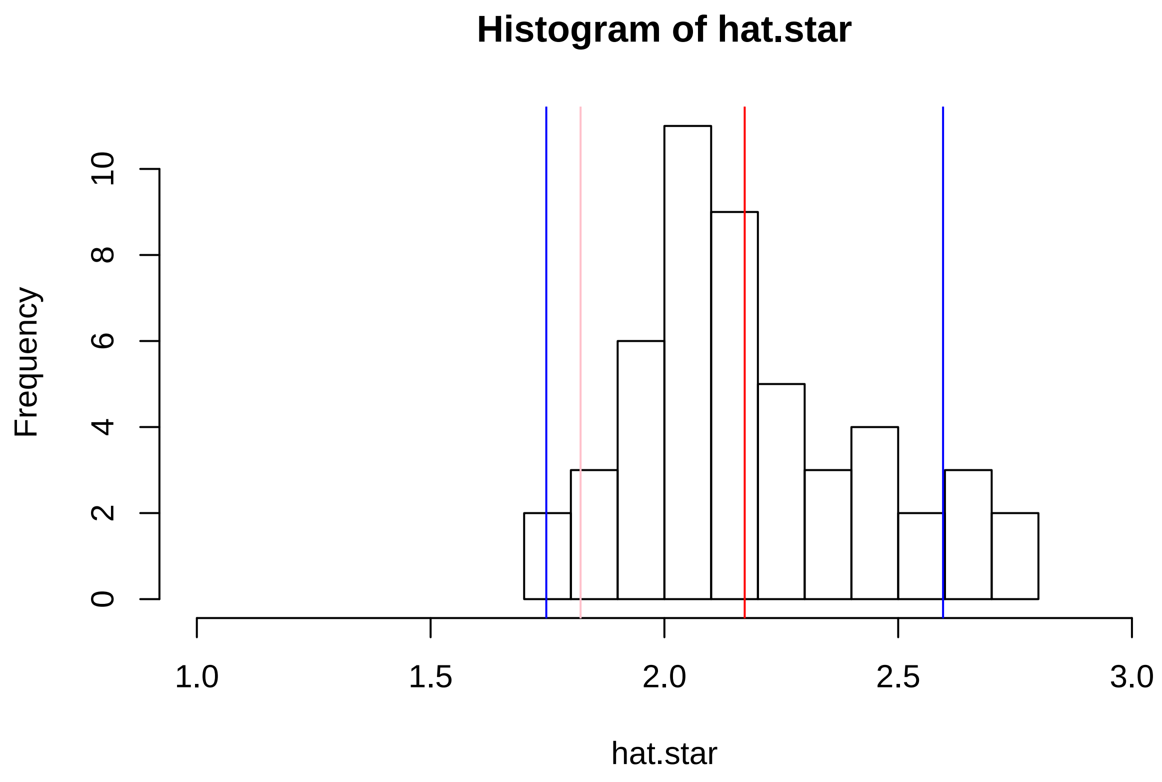


We'll now use the empirical quantiles of this simil-T distribution to estimate the CIs.

```
lower.q <- quantile(hat.star.T, probs = 0.025)*sd.hat.tm
upper.q <- quantile(hat.star.T, probs = 0.975)*sd.hat.tm
lower <- hat.tm + lower.q
upper <- hat.tm - lower.q
c(lower,upper)
```

```
## [1] 1.747416 2.595989
```

```
hist(hat.star, xlim = c(1,3))
abline(v=lower, col="blue")
abline(v=upper, col="blue")
abline(v=hat.tm, col="red")
abline(v=true.tm, col="pink")
```



And finally, let's check with the R results:

```
boot.res <- boot(data=small.sample,statistic=tm.var, R=50, sim="ordinary")
boot.ci(boot.res, conf=0.95, type="stud", index = c(1,2))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 50 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.res, conf = 0.95, type = "stud", index = c(1,
##     2))
##
## Intervals :
## Level    Studentized
## 95%   ( 1.807,  2.628 )
## Calculations and Intervals on Original Scale
## Some studentized intervals may be unstable
```

Simulations

Okay, as you have probably noticed, the bootstrap CI are variable. We'll now study their covarage exploiting this variability with a simulation.

```
nsim<-50
Bsim<-50
get.coverage <- function(ci){
  if(true.tm < ci[1]){return(c(1,0))}
  if(true.tm > ci[2]){return(c(0,1))}
```

```r
  return(c(0,0))
}
```

```r
coverages <- matrix(nrow=nsim,ncol=2*4) #each coverage has 2 columns
names.covrg<- names(coverages)<-c("norm.low","norm.high",
                    "basic.low","basic.high",
                    "percent.low","percent.high",
                    "student.low","student.high")
for(s in 1:nsim){
  new.sample <- rgamma(n,shape=2,rate=1)
  boot.res <- boot(data=new.sample,statistic=tm.var, R=Bsim,
                  sim="ordinary",parallel = "multicore", ncpus=20)
  b.ci<- boot.ci(boot.res, conf=0.95, type = c("basic", "norm", "perc", "stud"),
                index = c(1,2))
  coverages[s,1:2] <- get.coverage(b.ci$normal[2:3])
  coverages[s,3:4] <- get.coverage(b.ci$basic[4:5])
  coverages[s,5:6] <- get.coverage(b.ci$percent[4:5])
  coverages[s,7:8] <- get.coverage(b.ci$student[4:5])
}

coverages<- data.frame(coverages)
names(coverages) <- names.covrg
```

Let's look at the results of the simulation.

```r
#first: a check on the output
head(coverages)
```

```
##   norm.low norm.high basic.low basic.high percent.low percent.high student.low
## 1        0         0         0          0           0            0           0
## 2        0         0         0          0           0            0           0
## 3        0         0         1          0           0            0           0
## 4        0         0         0          1           0            0           0
## 5        0         0         0          0           0            0           0
## 6        0         0         0          0           0            0           0
##   student.high
## 1            0
## 2            0
## 3            0
## 4            0
## 5            0
## 6            0
```

```r
coverages.frac <- colMeans(coverages)
coverages.frac
```

```
##     norm.low    norm.high    basic.low   basic.high  percent.low percent.high
##         0.02         0.04         0.02         0.08         0.04         0.04
##  student.low student.high
##         0.02         0.02
```

Let's now simulate for multiple sample sizes, with more simulations.

```r
nsim <- 200
Bsim <- 200
ns <- c(10,50,100,500,1000)
coverages.ns <- matrix(nrow=length(ns),ncol=2*4) #each coverage has 2 columns
```

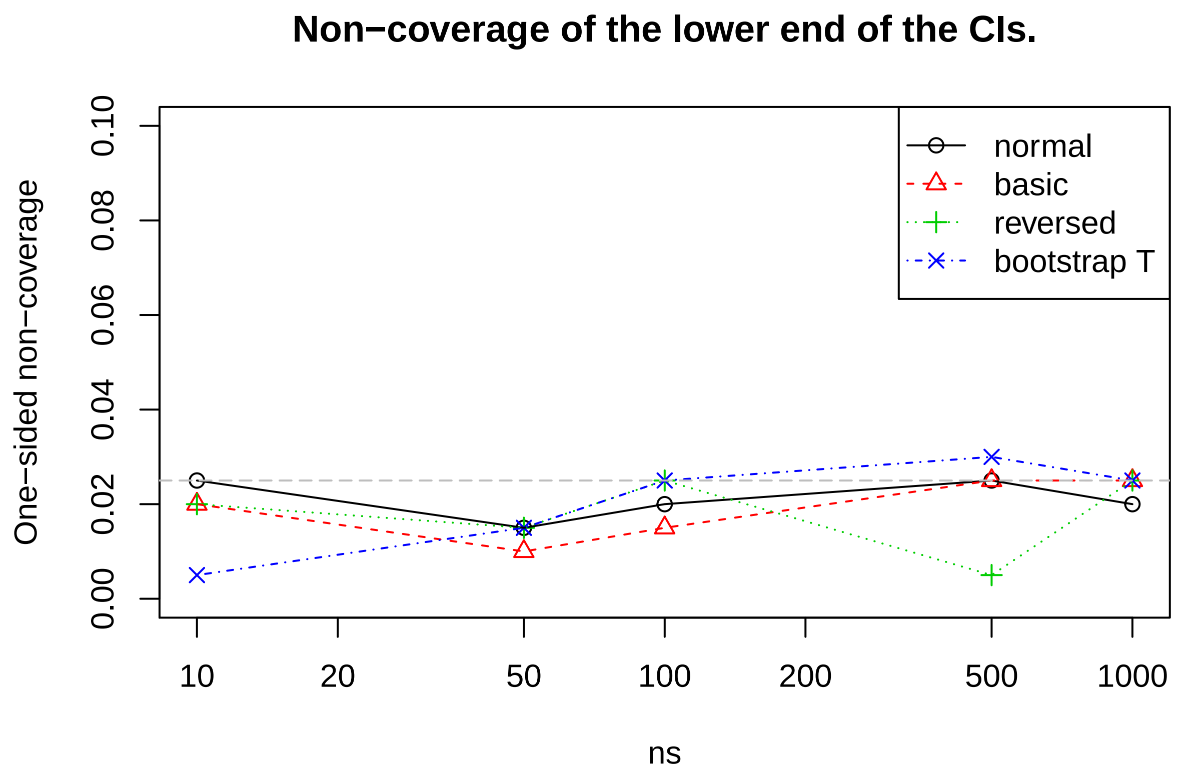
```r
for(i in 1:length(ns)){
  n <- ns[i]
  coverages <- matrix(nrow=nsim,ncol=2*4) #each coverage has 2 columns
  for(s in 1:nsim){
    new.sample <- rgamma(n,shape=2,rate=1)
    boot.res <- boot(data=new.sample,statistic=tm.var, R=Bsim,
                     sim="ordinary",parallel = "multicore", ncpus=20)
    b.ci<- boot.ci(boot.res, conf=0.95, type = c("basic", "norm", "perc", "stud"),
                   index = c(1,2))
    coverages[s,1:2] <- get.coverage(b.ci$normal[2:3])
    coverages[s,3:4] <- get.coverage(b.ci$basic[4:5])
    coverages[s,5:6] <- get.coverage(b.ci$percent[4:5])
    coverages[s,7:8] <- get.coverage(b.ci$student[4:5])
  }
  coverages.ns[i,] <- colMeans(coverages)
}

coverages.ns<- data.frame(coverages.ns, row.names=ns)
names(coverages.ns) <- names.covrg
head(coverages.ns)
```

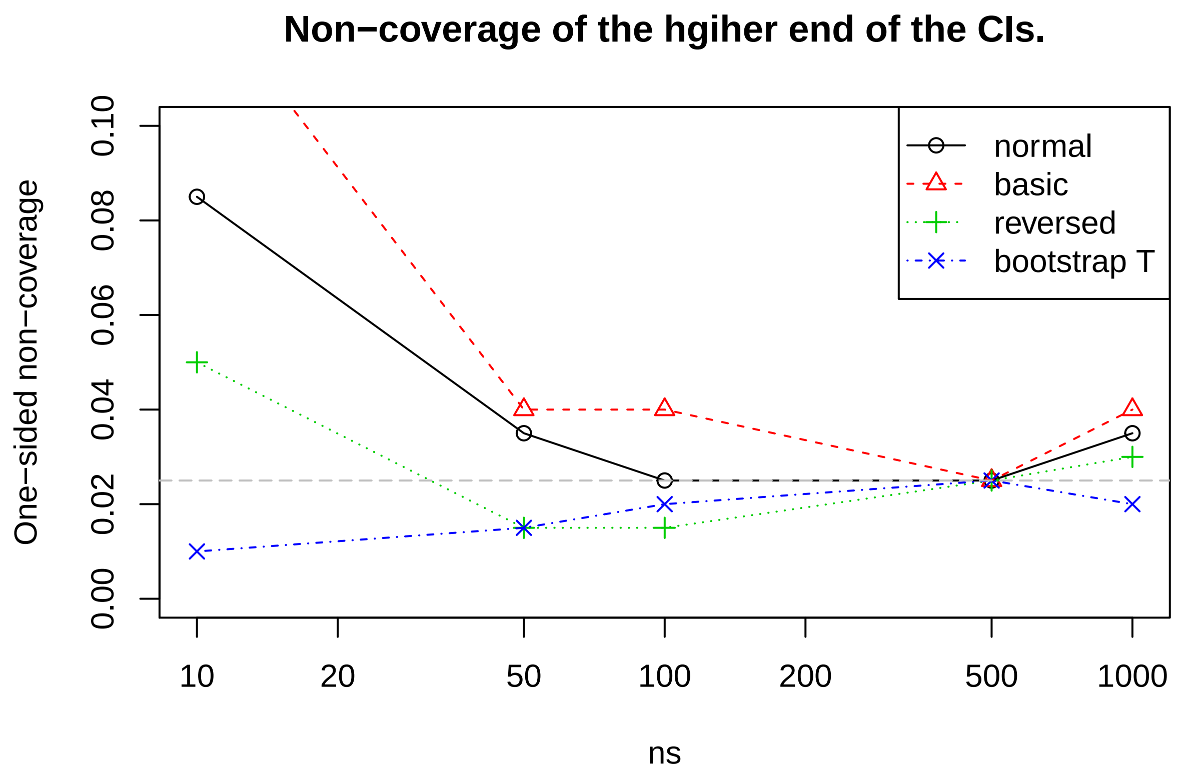
```
##      norm.low norm.high basic.low basic.high percent.low percent.high
## 10      0.025     0.085     0.020      0.130       0.020        0.050
## 50      0.015     0.035     0.010      0.040       0.015        0.015
## 100     0.020     0.025     0.015      0.040       0.025        0.015
## 500     0.025     0.025     0.025      0.025       0.005        0.025
## 1000    0.020     0.035     0.025      0.040       0.025        0.030
##      student.low student.high
## 10         0.005        0.010
## 50         0.015        0.015
## 100        0.025        0.020
## 500        0.030        0.025
## 1000       0.025        0.020
```

Let’s visualize the results.

```r
plot(norm.low~ ns, data = coverages.ns, col = 1, pch = 1, ylim = c(0, 0.1),
     log = "x", ylab = "One-sided non-coverage",
     main = "Non-coverage of the lower end of the CIs.")
points(basic.low ~ ns, data = coverages.ns, col = 2, pch = 2, xlog = TRUE)
points(percent.low ~ ns, data = coverages.ns, col = 3, pch = 3, xlog = TRUE)
points(student.low ~ ns, data = coverages.ns, col = 4, pch = 4, xlog = TRUE)
lines(norm.low ~ ns, data = coverages.ns, col = 1, lty = 1, xlog = TRUE)
lines(basic.low ~ ns, data = coverages.ns, col = 2, lty = 2, xlog = TRUE)
lines(percent.low ~ ns, data = coverages.ns, col = 3, lty = 3, xlog = TRUE)
lines(student.low ~ ns, data = coverages.ns, col = 4, lty = 4, xlog = TRUE)
abline(h = 0.025, lty = 5, col="gray")
legend("topright", legend = c("normal","basic", "reversed", "bootstrap T"),
       pch = 1:4, lty = 1:4, col = 1:4)
```



```
plot(norm.high~ ns, data = coverages.ns, col = 1, pch = 1, ylim = c(0, 0.1),
     log = "x", ylab = "One-sided non-coverage",
     main = "Non-coverage of the hgiher end of the CIs.")
points(basic.high ~ ns, data = coverages.ns, col = 2, pch = 2, xlog = TRUE)
points(percent.high ~ ns, data = coverages.ns, col = 3, pch = 3, xlog = TRUE)
points(student.high ~ ns, data = coverages.ns, col = 4, pch = 4, xlog = TRUE)
lines(norm.high ~ ns, data = coverages.ns, col = 1, lty = 1, xlog = TRUE)
lines(basic.high ~ ns, data = coverages.ns, col = 2, lty = 2, xlog = TRUE)
lines(percent.high ~ ns, data = coverages.ns, col = 3, lty = 3, xlog = TRUE)
lines(student.high ~ ns, data = coverages.ns, col = 4, lty = 4, xlog = TRUE)
abline(h = 0.025, lty = 5, col="gray")
legend("topright", legend = c("normal","basic", "reversed", "bootstrap T"),
       pch = 1:4, lty = 1:4, col = 1:4)
```



Notice the difference between the two tested sides, which could be due to asymmetry of the shape of the gamma distribution.