Bootstrap Confidence Intervals

Oliver d’Pug

Bootstrapping.QMD and its output contain information on how to compute/generate bootstrap samples and compute the bootstrap distribution of a statistic. The various approaches that are presented allow the computation of bias and standard error estimates. In what follows, we make use of the **boot** package to fit the bootstrap distributions, and from the distribution a number of different confidence intervals for the parameter of interest.

## The **boot** Package’s Bootstrap Distribution

The **boot** package can be obtained from CRAN. Once loaded, it is easy to use the ***boot*** *function* to create bootstrap distributions for a statistic that we have defined. Below we create functions that **boot** can call.;

### Estimate for

In Rao-Blackwell and Lehmann-Scheffe we saw that it is possible to update a statistic to make it unbiased. Consider a random variables and a statistic . Since we wish to find an unbiased estimator.

If we let then then Rao-Blackwell shows that is an unbiased estimator of . To confirm this we can look at the bootstrap distributions of , , and . We start by defining a function that computes the statistics within the **boot** function.

normal\_mystat = function(d, i){  
 n = length(i) ### Not used and a waste of time  
   
 T = sum(d[i]) ### Sum\_{i=1}^n X\_i  
 U = d[i[1]] ### X\_1  
 V = mean(d[i]) ### \overline{X}  
   
 c(T, U, V) ### Return the list  
 }

We generate a number of observations from a normal distribution of our choice.

### Set the seed to 47 to replicate output.   
 set.seed(47) ### Comment this line out to use the clock.  
  
 n = 100  
 mu = 3  
 s = 5  
   
 normal\_data = rnorm(n, mu, s)  
   
 write.csv(normal\_data, "normal\_data.csv")

We can now use the **normal\_mystat** function to get the bootstrap distributions.

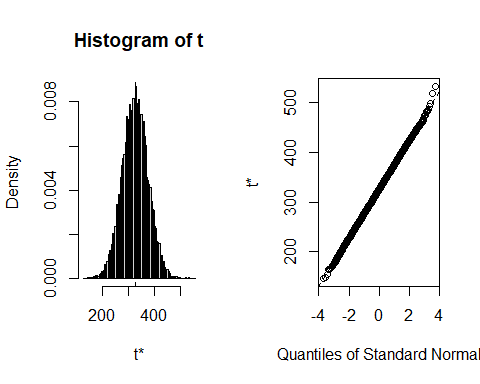
### Use the boot function to run the bootstrap  
 normal\_boot = boot(normal\_data, normal\_mystat, R=9999)  
  
 ### Check the behavior of the statistics  
 normal\_boot

ORDINARY NONPARAMETRIC BOOTSTRAP  
  
  
Call:  
boot(data = normal\_data, statistic = normal\_mystat, R = 9999)  
  
  
Bootstrap Statistics :  
 original bias std. error  
t1\* 325.744525 -0.423393654 48.9468515  
t2\* 12.973482 -9.701413166 4.8954405  
t3\* 3.257445 -0.004233937 0.4894685

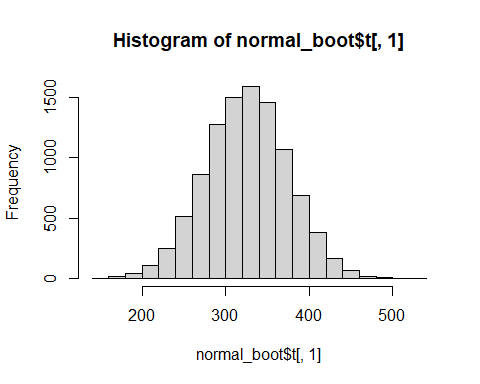
summary(normal\_boot)

Number of bootstrap replications R = 9999   
 original bootBias bootSE bootMed  
1 325.7445 -0.4233937 48.94685 325.7732  
2 12.9735 -9.7014132 4.89544 3.1980  
3 3.2574 -0.0042339 0.48947 3.2577

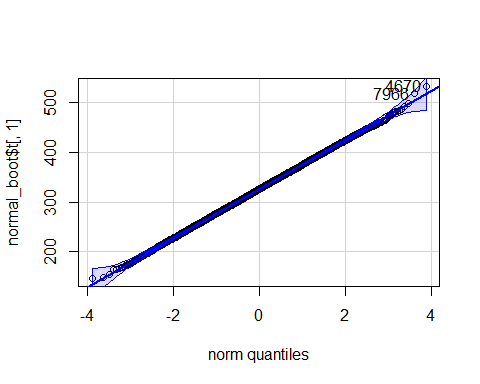
plot(normal\_boot)



hist(normal\_boot$t[,1])

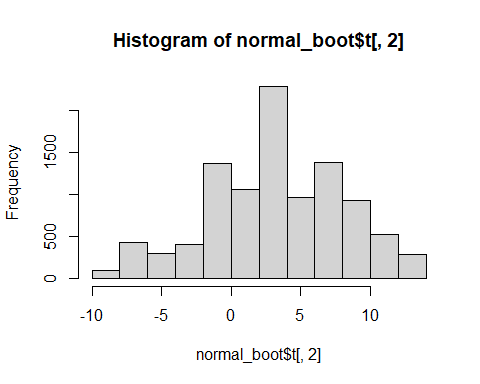


qqPlot(normal\_boot$t[,1], distribution="norm")

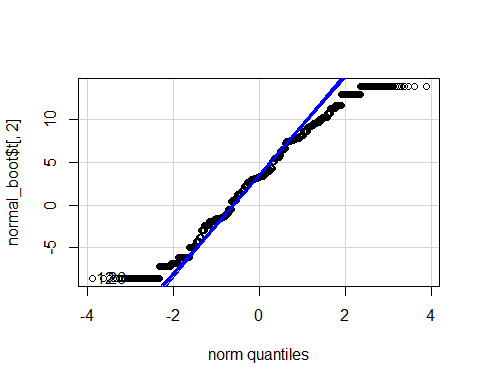


[1] 4670 7966

hist(normal\_boot$t[,2])

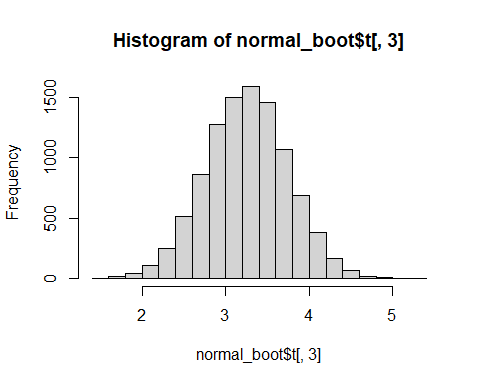


qqPlot(normal\_boot$t[,2], distribution="norm")

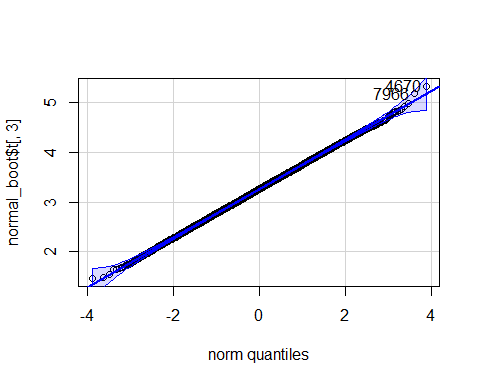


[1] 12 26

hist(normal\_boot$t[,3])



qqPlot(normal\_boot$t[,3], distribution="norm")



[1] 4670 7966

Note that the distributions of the sum and mean are both clearly normal. On the other hand, while normal by definition, a single observation appears to be less normal.

quantile(normal\_boot$t[,3], c(0.025, 0.975))

2.5% 97.5%   
2.292393 4.205230

args(boot.ci)

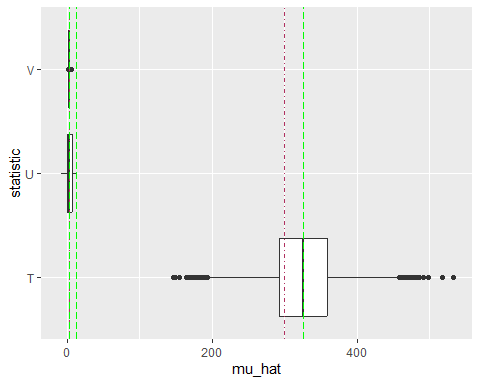
function (boot.out, conf = 0.95, type = "all", index = 1L:min(2L,   
 length(boot.out$t0)), var.t0 = NULL, var.t = NULL, t0 = NULL,   
 t = NULL, L = NULL, h = function(t) t, hdot = function(t) rep(1,   
 length(t)), hinv = function(t) t, ...)   
NULL

boot.ci(normal\_boot, type="all", index=3)

Warning in boot.ci(normal\_boot, type = "all", index = 3): bootstrap variances  
needed for studentized intervals

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
Based on 9999 bootstrap replicates  
  
CALL :   
boot.ci(boot.out = normal\_boot, type = "all", index = 3)  
  
Intervals :   
Level Normal Basic   
95% ( 2.302, 4.221 ) ( 2.310, 4.223 )   
  
Level Percentile BCa   
95% ( 2.292, 4.205 ) ( 2.287, 4.200 )   
Calculations and Intervals on Original Scale

x = as.data.frame(normal\_boot$t)  
 colnames(x) = c("T", "U", "V")  
 x |>  
 pivot\_longer(cols =c("T","U","V"),   
 names\_to = "statistic",  
 values\_to = "mu\_hat",   
 values\_drop\_na = TRUE) |>  
 ggplot(aes(y = statistic, x = mu\_hat)) +  
 geom\_boxplot() +  
 geom\_vline(xintercept=c(mu, n\*mu), color="maroon", lty=4) +  
 geom\_vline(xintercept=c(normal\_boot$t0), color="green", lty=5)



We can check that the theoretical and empirical bias and standard errors are similar.

normal\_bias = as.data.frame(cbind(summary(normal\_boot)$original - c(n\*mu, mu, mu),   
 summary(normal\_boot)$bootBias,   
 c(0,0,0)))  
 colnames(normal\_bias) = c("Original", "Bootstrap", "Theoretical")  
 rownames(normal\_bias) = c("Sum","X\_1","Xbar")  
 normal\_bias

Original Bootstrap Theoretical  
Sum 25.7445253 -0.423393654 0  
X\_1 9.9734817 -9.701413166 0  
Xbar 0.2574453 -0.004233937 0

normal\_ses = as.data.frame(cbind(summary(normal\_boot)$bootSE, c(sqrt(n)\*s, s, s/sqrt(n))))  
 colnames(normal\_ses) = c("Bootstrap", "Theoretical")  
 rownames(normal\_ses) = c("Sum","X\_1","Xbar")  
 normal\_ses

Bootstrap Theoretical  
Sum 48.9468515 50.0  
X\_1 4.8954405 5.0  
Xbar 0.4894685 0.5

Note that **boot.ci** uses its own quantile function rather than relying upon the base **quantile** function. Minor interpolation differences between interval estimates are common.

### Estimate for

As we saw earlier, Lehmann-Scheffe II can be used to get an estimator for when we have random variables and a statistic . Since is based upon a minimal sufficient statistic and can be shown to be complete, we need only to find a constant that makes our new statistic have expectation . We note that has expectation

So, is UMVUE for .

We can check the behavior of the unbiased estimators and using bootstrapping.

uniform\_mystat = function(d, i){  
 n = length(i) ### Require for V  
   
 T = max(d[i]) ### X\_([n]) is biased  
 U = 2 \* mean(d[i]) ### 2 \* \overline{X}  
 V = (n+1) \* T / n ### (n+1) X\_{[n]} / n  
   
 c(T, U, V) ### Return the list  
 }

We generate a number of observations from a distribution of our choice.

### Set the seed to 47 to replicate output.   
 set.seed(47) ### Comment this line out to use the clock.  
  
 n = 100  
 theta = 5  
   
 uniform\_data = runif(n, 0, theta)  
   
 write.csv(uniform\_data, "uniform\_data.csv")

We can now use the **uniform\_mystat** function to get the bootstrap distributions.

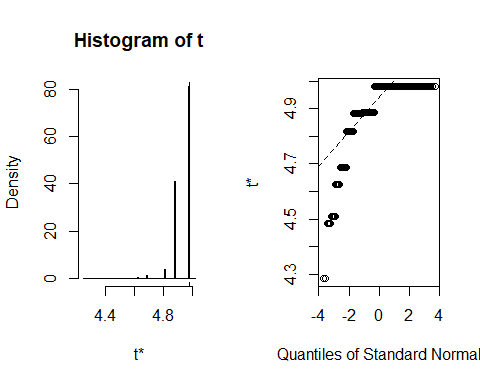
### Use the boot function to run the bootstrap  
 uniform\_boot = boot(uniform\_data, uniform\_mystat, R=9999)  
  
 ### Check the behavior of the statistics  
 uniform\_boot

ORDINARY NONPARAMETRIC BOOTSTRAP  
  
  
Call:  
boot(data = uniform\_data, statistic = uniform\_mystat, R = 9999)  
  
  
Bootstrap Statistics :  
 original bias std. error  
t1\* 4.980499 -0.0409126659 0.06218675  
t2\* 4.730612 0.0007536833 0.28421486  
t3\* 5.030304 -0.0413217926 0.06280862

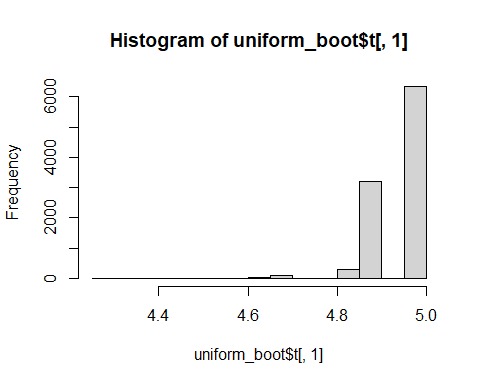
summary(uniform\_boot)

Number of bootstrap replications R = 9999   
 original bootBias bootSE bootMed  
1 4.9805 -0.04091267 0.062187 4.9805  
2 4.7306 0.00075368 0.284215 4.7297  
3 5.0303 -0.04132179 0.062809 5.0303

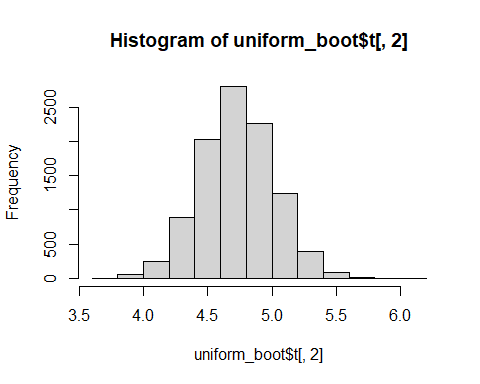
plot(uniform\_boot)



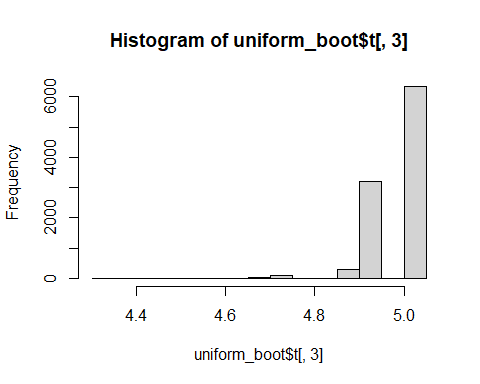
hist(uniform\_boot$t[,1])



hist(uniform\_boot$t[,2])



hist(uniform\_boot$t[,3])



Note that the distributions of the sum and mean are both clearly normal. On the other hand, while normal by definition, a single observation appears to be less normal.

quantile(uniform\_boot$t[,3], c(0.025, 0.975))

2.5% 97.5%   
4.863812 5.030304

args(boot.ci)

function (boot.out, conf = 0.95, type = "all", index = 1L:min(2L,   
 length(boot.out$t0)), var.t0 = NULL, var.t = NULL, t0 = NULL,   
 t = NULL, L = NULL, h = function(t) t, hdot = function(t) rep(1,   
 length(t)), hinv = function(t) t, ...)   
NULL

uniform\_boot.ci = boot.ci(uniform\_boot, index=3)

Warning in boot.ci(uniform\_boot, index = 3): bootstrap variances needed for  
studentized intervals

uniform\_boot.ci

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
Based on 9999 bootstrap replicates  
  
CALL :   
boot.ci(boot.out = uniform\_boot, index = 3)  
  
Intervals :   
Level Normal Basic   
95% ( 4.949, 5.195 ) ( 5.030, 5.197 )   
  
Level Percentile BCa   
95% ( 4.864, 5.030 ) ( 4.864, 5.030 )   
Calculations and Intervals on Original Scale

x = as.data.frame(uniform\_boot$t)  
 colnames(x) = c("T", "U", "V")  
 x |>  
 pivot\_longer(cols =c("T","U","V"),   
 names\_to = "statistic",  
 values\_to = "theta\_hat",   
 values\_drop\_na = TRUE) |>  
 ggplot(aes(y = statistic, x = theta\_hat)) +  
 geom\_boxplot() +  
 geom\_vline(xintercept=theta, color="maroon", lty=2)

