Namburu_Setu

Fisher proposed a method for calculating a confidence interval for the Pearson Correlation Coefficient. This method involves transformation of r, the sample Pearson Correlation Coefficient. A z-score is calculated and confidence interval is constructed. This confidence interval is then transformed back to the original scale of measurement. This method is explained in the links:

http://www2.sas.com/proceedings/sugi31/170-31.pdf

http://onlinestatbook.com/2/estimation/correlation_ci.html

Use the data provided and construct the data frame “test” with the code below. The data frame contains test results for 49 students on two standardized tests. Each student took both tests. Do not change the order of the two test entries or the matching per student will not be correct

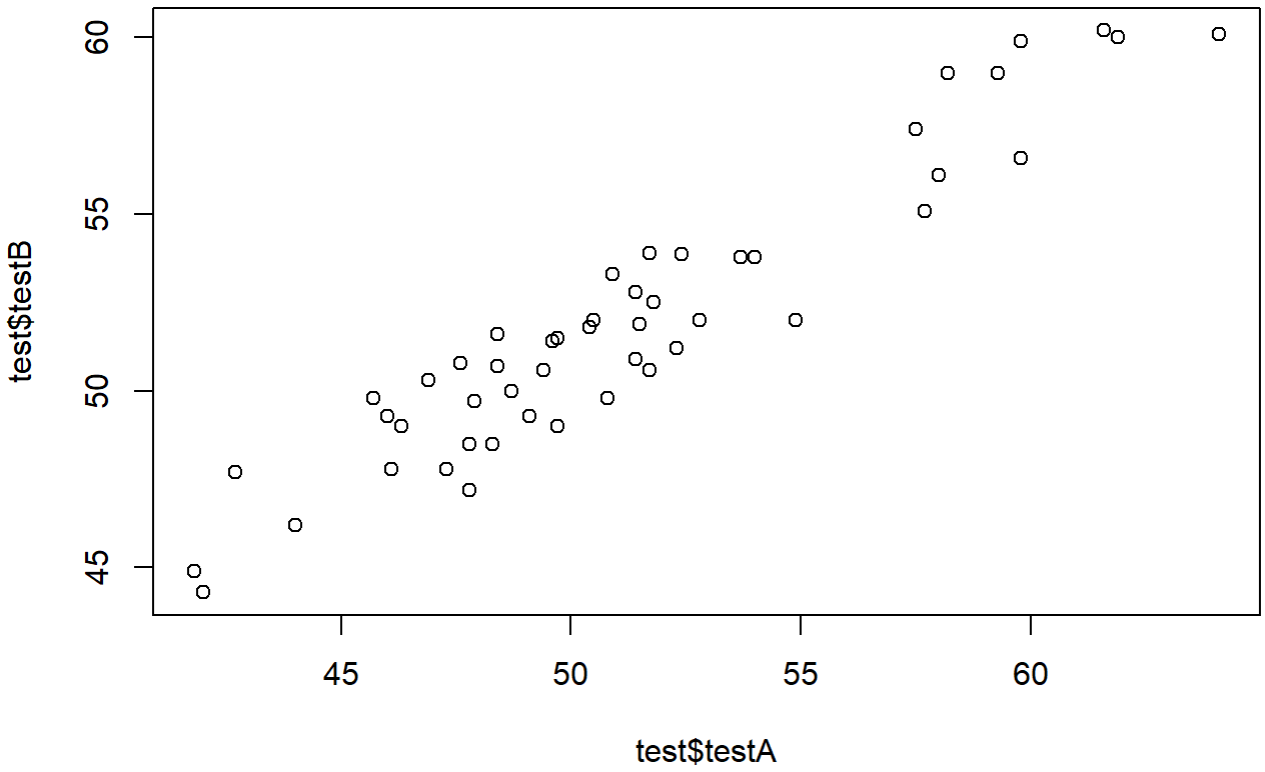
```
testA <- c(58,49.7,51.4,51.8,57.5,52.4,47.8,45.7,51.7,46,50.4,61.9,49.6,61.6,54,54.9,49.7,
           47.9,59.8,52.3,48.4,49.1,53.7,48.4,47.6,50.8,58.2,59.8,42.7,47.8,51.4,50.9,49.4,
           64.1,51.7,48.7,48.3,46.1,47.3,57.7,41.8,51.5,46.9,42,50.5,46.3,44,59.3,52.8)
testB <- c(56.1,51.5,52.8,52.5,57.4,53.86,48.5,49.8,53.9,49.3,51.8,60,51.4,60.2,53.8,52,
           49,49.7,59.9,51.2,51.6,49.3,53.8,50.7,50.8,49.8,59,56.6,47.7,47.2,50.9,53.3,
           50.6,60.1,50.6,50,48.5,47.8,47.8,55.1,44.9,51.9,50.3,44.3,52,49,46.2,59,52)

test <- as.data.frame(cbind(testA,testB))

str(test)
```

```
## 'data.frame':    49 obs. of  2 variables:
##  $ testA: num  58 49.7 51.4 51.8 57.5 52.4 47.8 45.7 51.7 46 ...
##  $ testB: num  56.1 51.5 52.8 52.5 57.4 ...
```

```
summary(test)
```

```
##      testA           testB      
##  Min.   :41.80   Min.   :44.30  
##  1st Qu.:47.80   1st Qu.:49.30  
##  Median :50.50   Median :51.40  
##  Mean   :51.25   Mean   :51.95  
##  3rd Qu.:53.70   3rd Qu.:53.80  
##  Max.   :64.10   Max.   :60.20
```

```
plot(test$testA,test$testB)
```



Part #1 (3 points)

Determine a 95% confidence interval for the Pearson Correlation Coefficient of the data in “test” using Fisher’s method. Present the code and the confidence interval for rho, the Pearson Correlation Coefficient. Calculations can be simplified using *tanh()* and *atanh()*. Also, submit the data to *cor.test()* and present those results as well.

```
r <- cor(test$testA,test$testB)
r
```

```
## [1] 0.9492478
```

```
fisherz <- 0.5*log((1+r)/(1-r))
sigmaz <- 1/sqrt(49-3)
zl.95 <- fisherz - qnorm(0.975,0,1,lower.tail = TRUE)*sigmaz
zu.95 <- fisherz + qnorm(0.975,0,1,lower.tail = TRUE)*sigmaz

rl.95 <- tanh(zl.95)
ru.95 <- tanh(zu.95)

cat("95% confidence interval for correlation coefficient:", c(rl.95,ru.95),'\n')
```

```
## 95% confidence interval for correlation coefficient: 0.9113003 0.9712052
```

```
cor.test(test$testA,testB)
```

```
##
##  Pearson's product-moment correlation
##
## data:  test$testA and testB
## t = 20.69, df = 47, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9113003 0.9712052
## sample estimates:
##       cor
## 0.9492478
```

Part #2 (5 points)

Bootstrapping can be used to arrive at an estimated confidence interval. The process involves resampling with replacement of rows from "test." The first step is to randomly sample with replacement the 49 rows of "test". Each sample will consist of 49 rows for which a sample correlation coefficient is calculated. For this purpose, the function *sample.int()* may be used to determine a sample of row numbers to be used. The function *cor()* should be used to determine the sample correlation coefficient. This step is repeated 10,000 times resulting in 10,000 sample correlation coefficients. The 10,000 calculated sample correlation coefficients are written to a vector. *quantile()* is passed this vector to calculate the 2.5% (0.025) and 97.5% (0.975) quantiles which determines the 95% Percentile Bootstrap confidence interval.

Refer to the course site library reserves and read the listing for Section 9.5 of "Mathematical Statistics with Resampling and R," by Chihara and Hesterberg.

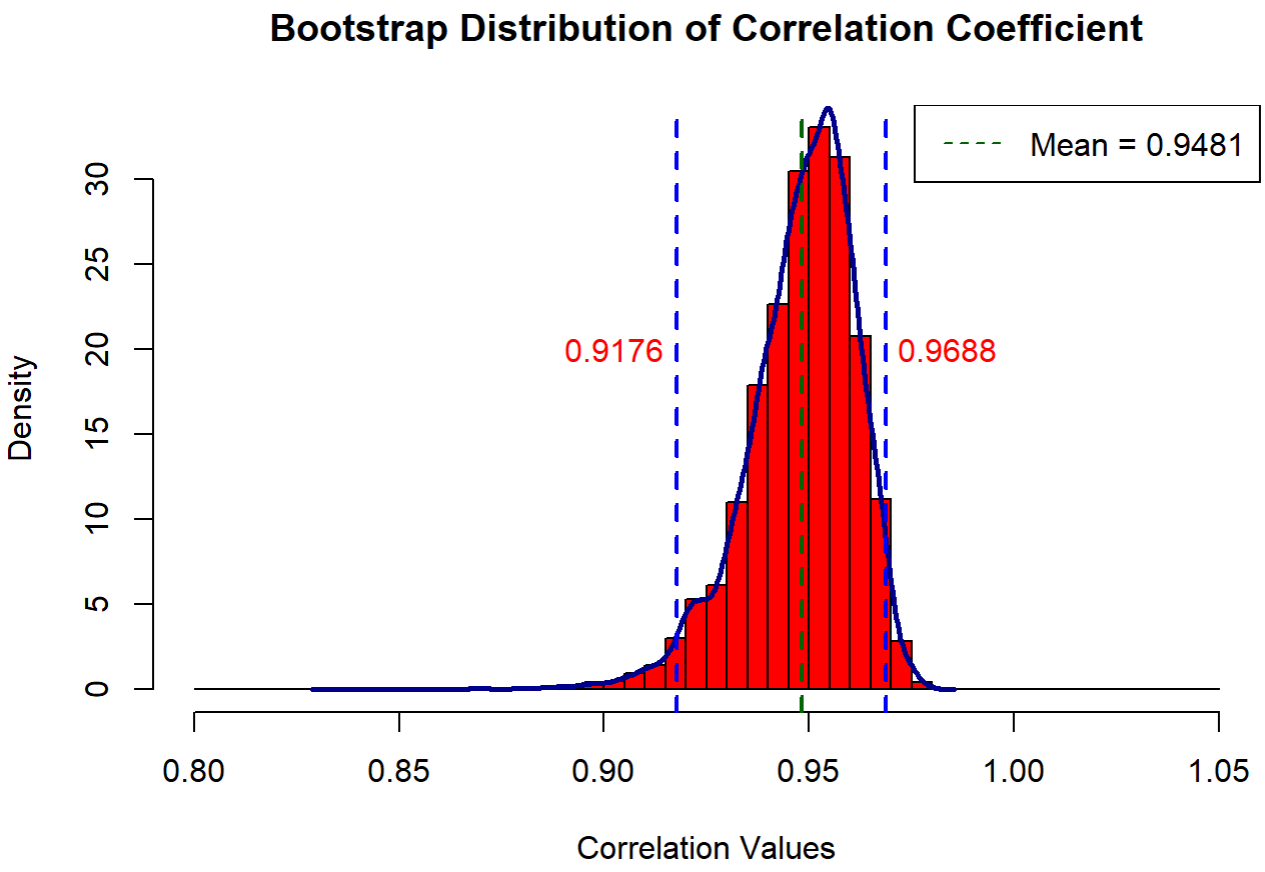
You will write code which does this work. Use *set.seed(123)*. A "for" loop may be used to repeat the sampling and correlation coefficient calculations. Plot a histogram and report a two-sided 95% confidence interval.

```
set.seed(123)
N <- 10000
n <- 49

get.cor<-function(data){
  return(cor(data$testA,data$testB))
}

m.corr <- replicate(N,get.cor(test[sample.int(n,n,replace = TRUE),]))
summary(m.corr)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8343  0.9407  0.9499  0.9481  0.9575  0.9800
```

```
cells <- seq(0.8,1.05, by = 0.005)
hist(m.corr, main = "Bootstrap Distribution of Correlation Coefficient",breaks = cells,
     freq  = FALSE, col = "red", xlab = "Correlation Values")
lines(density(m.corr), lwd=2.5, col = "darkblue")
abline(v= quantile(m.corr, probs = 0.025), col = "blue", lwd = 2, lty = 2)
abline(v= quantile(m.corr, probs = 0.975), col = "blue", lwd = 2, lty = 2)
abline(v= mean(m.corr), col = "darkgreen", lwd = 2, lty = 2)
```

```
text(quantile(m.corr, probs = 0.025)-0.015,20,round(quantile(m.corr, probs = 0.025),4), col =
2)
text(quantile(m.corr, probs = 0.975)+0.015,20,round(quantile(m.corr, probs = 0.975),4), col =
2)
legend ( "topright",legend = c("Mean = 0.9481"), col = c("darkgreen"),lty=c(2))
```




```
#legend ( "topright",legend =  round(mean(m.corr),4), col = c("darkgreen"),lty=c(2))


print("95% confidence interval for bootstrapped correlation coefficient: ")
```

```
## [1] "95% confidence interval for bootstrapped correlation coefficient: "
```

```
round(c(quantile(m.corr, probs = c(0.025)), quantile(m.corr, probs = c(0.975))), digits = 4)
```

```
##   2.5%  97.5%
## 0.9176 0.9688
```

How do bootstrapping results compare to the results from Part 1?

Answer: As observed from bootstrapping summary statistics (mean is not equal to median), the distribution is negatively skewed as observed through histogram. This indicates bias towards zero (which indicates the mean of boortstrapped distribution is smaller than the correltion coefficient of original data), but the result is reasonable as the coefficient is near 1. Bootstrapping didn't offer much in terms of accuracy compared to method in part 1,

but it has helped visually to observe the variation in sampling distribution. Good!

Part #3 (7 points)

Bootstrapping can also be used to arrive at confidence intervals for regression coefficients. This process is similar to the process in Part #2. Using the current data frame, rows of “test” are randomly sampled with replacement. Each sample is passed to *lm() and the coefficients extracted. Section 9.5 of Chihara and Hesterberg give an example R script showing how this may be accomplished.

Write code using “test” to produce histograms and 95% two-sided bootstrap confidence intervals for the intercept and slope of a simple linear regression. Please keep *set.seed(123)*. A “for” loop can be written to sample via *sample.int()*, a linear regression model fit via *lm()* and the coefficients extracted via *coef()*. Note that we pass our fitted model object to *coef()* to return the coefficients; e.g. *coef(model)[1]* should return the intercept, *coef(model)[2]* the slope.

Present two histograms, one for bootstrap intercept results, and one for the bootstrap slope results showing the 2.5% and 97.5% quantiles on each. In addition, show the location on the histograms of the estimated intercept and slope of test using lm() without bootstrapping.

Lastly, generate a scatter plot of the estimated bootstrap slopes versus the estimated bootstrap intercepts. There will be 10,000 points appearing in this plot, one for each bootstrap sample. Place the intercept on the x-axis and the slope on the y-axis.

```
set.seed(123)

N <- 10000
beta.boot <- numeric(N)
alpha.boot <- numeric(N)
n <- 49

for (i in 1:N)
{
  index <- sample.int(n,n, replace = TRUE)
  test1 <- test[index, ] # resampled data
  #recalculate linear model estimates
  tBoot.lm <- lm(testB ~ testA, data = test1)
  alpha.boot[i] <- coef(tBoot.lm)[1] # new intercept
  beta.boot[i] <- coef(tBoot.lm)[2] # new slope

}

print("Bootstrapped Intercept confidence Interval:")
```

```
## [1] "Bootstrapped Intercept confidence Interval:"
```

```
quantile(alpha.boot, c(.025,.975))
```

```
##    2.5%   97.5%
## 11.9528 18.9528
```

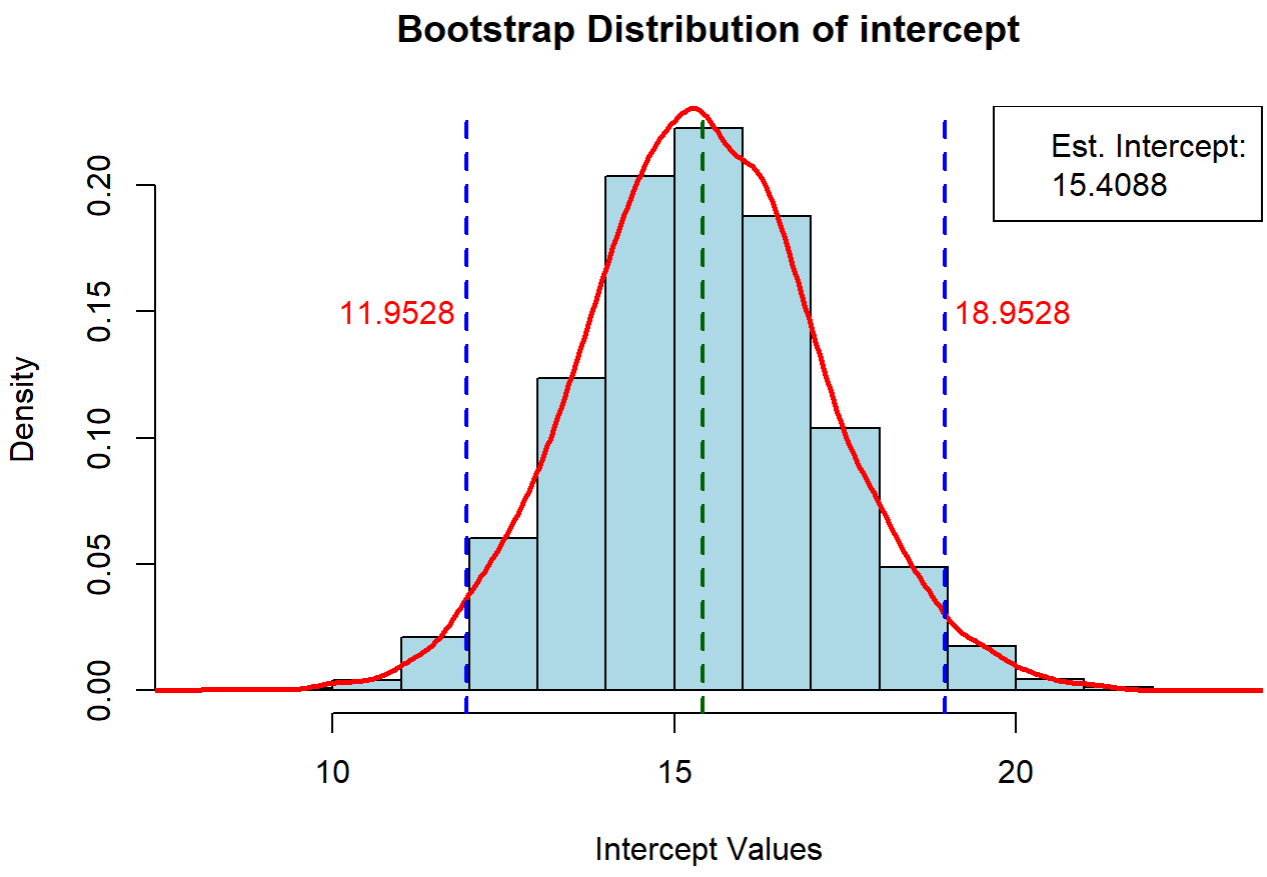
```
print("Bootstrapped Slope confidence Interval:")
```

```
## [1] "Bootstrapped Slope confidence Interval:"
```

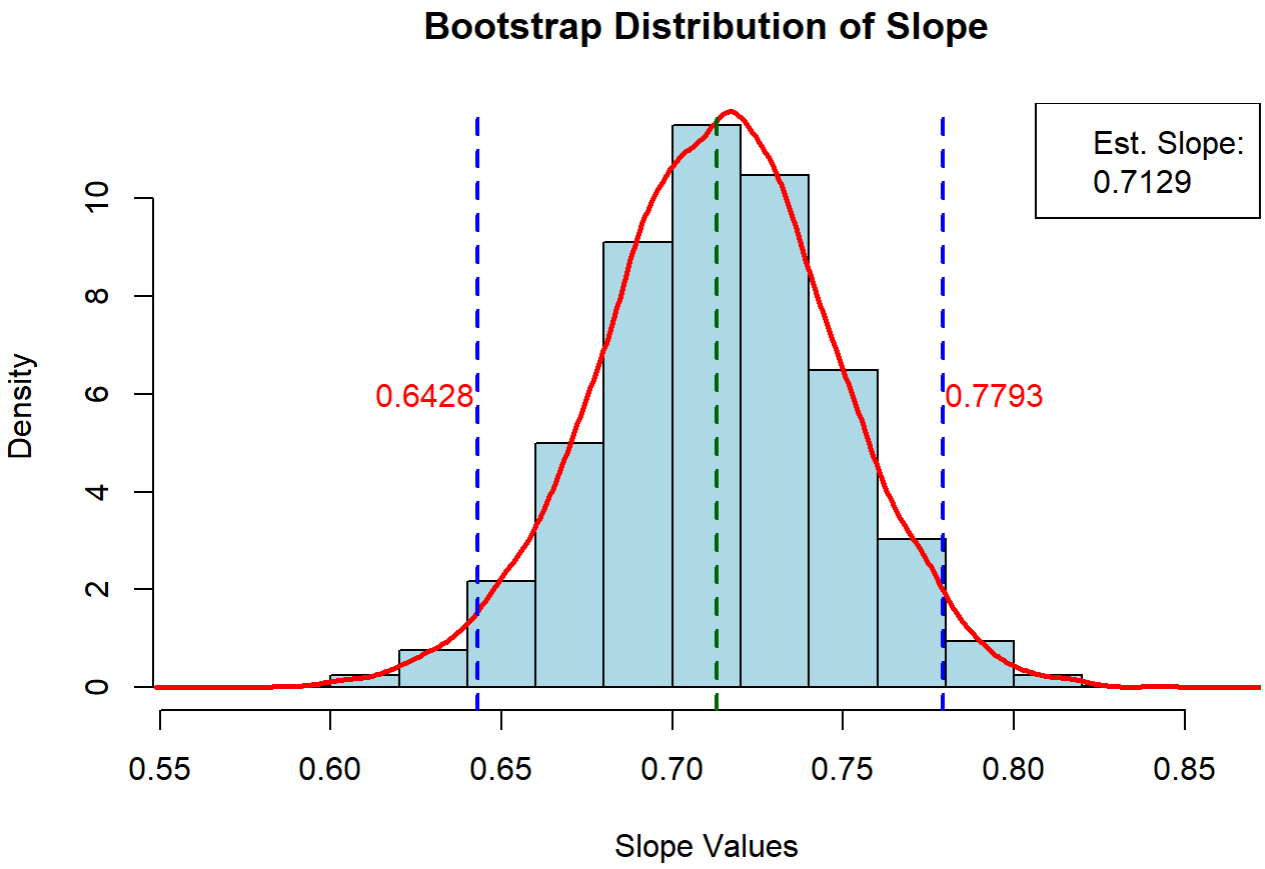
```
quantile(beta.boot, c(.025,.975))
```

```
##      2.5%     97.5%
## 0.6427790 0.7792631
```

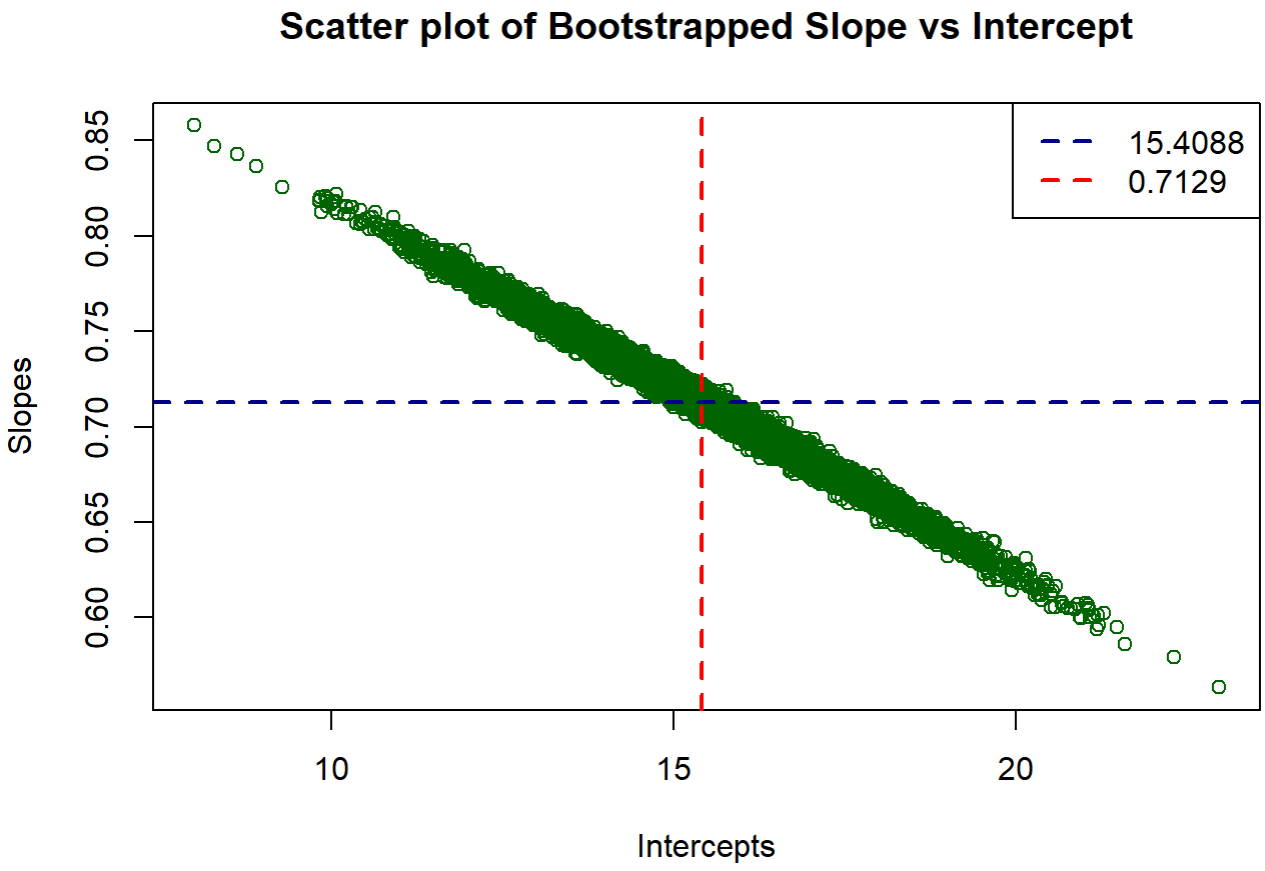
```
model <- lm(testB~testA, data = test)
lm.coef1 <- round(coef(model)[1],4)
lm.coef2 <- round(coef(model)[2],4)

ll<-round(quantile(alpha.boot,probs = 0.025),4)
ul<-round(quantile(alpha.boot,probs = 0.975),4)
hist(alpha.boot, main = "Bootstrap Distribution of intercept",
     freq  = FALSE, col = "lightblue", xlab = "Intercept Values")
lines(density(alpha.boot), lwd=2.5, col = "red")
abline(v= ll, col = "blue", lwd = 2, lty = 2)
abline(v= ul, col = "blue", lwd = 2, lty = 2)
abline(v = lm.coef1, col = "darkgreen",lty = 2, lwd=2)
text(ll-1,0.15,ll, col = 2)
text(ul+1,0.15,ul, col = 2)
legend ( "topright",legend = c("Est. Intercept:", lm.coef1))
```



```
ll<-round(quantile(beta.boot,probs = 0.025),4)
ul<-round(quantile(beta.boot,probs = 0.975),4)
hist(beta.boot, main = "Bootstrap Distribution of Slope",
     freq  = FALSE, col = "lightblue", xlab = "Slope Values")
lines(density(beta.boot), lwd=2.5, col = "red")
abline(v= ll, col = "blue", lwd = 2, lty = 2)
abline(v= ul, col = "blue", lwd = 2, lty = 2)
abline(v = lm.coef2, col = "darkgreen",lty = 2, lwd=2)
text(ll-0.015,6,ll, col = 2)
text(ul+0.015,6,ul, col = 2)
legend ( "topright",legend = c("Est. Slope:", lm.coef2))
```




```
df<-as.data.frame(cbind(alpha.boot,beta.boot))
plot(df$alpha.boot,df$beta.boot, main = "Scatter plot of Bootstrapped Slope vs Intercept", col
 = "darkgreen", xlab = "Intercepts", ylab = "Slopes")
abline(v = lm.coef1, col = "red", lw=2, lty=2)
abline(h = lm.coef2, col = "darkblue", lw=2,lty = 2)
legend ( "topright",legend = c(lm.coef1, lm.coef2), col = c("darkblue", "red"),lty=c(2,2),lw=2
)
```



What does the plot of the estimated bootstrap slopes versus the estimated bootstrap intercepts plot indicate about these estimates?

Answer: The plot indicates a negative correlation between estimated booststrap slopes and intercept values, i.e., slope and intercept are inversely related.

-1 point Why is this? We are constrained to one set of data. As the intercept goes up, the slope must go down and vice versa.

14 points