Statistical Models; Homework 5

Sourabh Prakash and Priyanshi Shah

2023-02-16

```
library(ggplot2)
```

Question1:

Write a function named bootLS(x, y, conf = 0.95, B = 1000) that fits a simple linear model explaining y in terms of x, and returns a studentized bootstrap confidence interval at the desired level based on the specified number of repeats for each coefficient vector. (If conf = 0.95, then each interval will have nominal level 0.95, so the the confidence is individual and not simultaneous, as is the case with the function confint.)

Function definition for bootLS(x,y,conf=0.95,B=1000) that fits a simple linear model y-x and returns studentized bootstrap confidence interval.

```
set.seed(2022)
#Defining the BootLS funcation
bootLS = function(x,y, conf = 0.95, B=1000){
  N = length(x)
  beta0_boot = rep(NA,N)
  beta1_boot = rep(NA,N)
  t0_boot = rep(NA,N)
  t1_boot = rep(NA,N)

  #fitting the model
  fit = lm(y~x)
  beta0 = fit$coefficients[1]
  beta1 = fit$coefficients[2]
  sebeta0=summary(fit )$coefficients[,2][1]
  sebeta1=summary(fit)$coefficients[,2][2]

  set.seed(241)
  #looping
  for (i in 1:B){
    indices = sample(1:N,N,replace=TRUE)
    x_boot = x[indices]
    y_boot = y[indices]
    fit_boot=lm(y_boot~x_boot)
    beta0_boot[i] = fit_boot$coefficients[1]
    beta1_boot[i] = fit_boot$coefficients[2]
    sebeta0_boot = summary(fit_boot)$coefficients[,2][1]
    sebeta1_boot = summary(fit_boot)$coefficients[,2][2]
    t0_boot[i]=(beta0_boot[i]-beta0)/(sebeta0_boot)
```

```
    t1_boot[i]=(beta1_boot[i]-beta1)/(sebeta1_boot)
  }

  c1 = ((1 - conf)/2)*100
  c2 = (100 - c1)
  boot_int = matrix(c(beta0 + quantile(t0_boot,c((1-conf)/2,(1+conf)/2))*sebeta0),ncol = 2)
  colnames(boot_int) = c(paste(c1,"%"),paste(c2, "%"))
  boot_slp = matrix(c(beta1 + quantile(t1_boot,c((1-conf)/2,(1+conf)/2))*sebeta1),ncol = 2)
  colnames(boot_slp) = c(paste(c1,"%"),paste(c2, "%"))
  return(c(boot_int,boot_slp))

}
```

```
#Loading the dataset
#setwd("D:/projects/Quarter-2/Stats_model")
load("04cars.rda")
intervals = bootLS(dat$Weight, dat$City_MPG)
#fitting the model
fit = lm(dat$City_MPG~dat$Weight)

temp = matrix(c(c(intervals[1],intervals[3]),c(intervals[2],intervals[4])), nrow= 2)
colnames(temp) = c("2.5%", "97.5%")
print(temp)
```

```
##              2.5%        97.5%
## [1,] 35.698294157 41.378853029
## [2,] -0.005927099 -0.004447527
```

```
#print(intervals)
confint(fit, level = 0.95)
```

```
##                    2.5 %       97.5 %
## (Intercept) 37.19932843 40.615740179
## dat$Weight  -0.00574151 -0.004803218
```

##Inference: Studentized bootstrap confidence intervals can be a more reliable and accurate method of estimating confidence intervals, especially when the assumptions of normality are violated or the sample size is small. However, it should be noted that bootstrap methods can be computationally intensive and may take longer to compute than normal-based methods.

Question2:

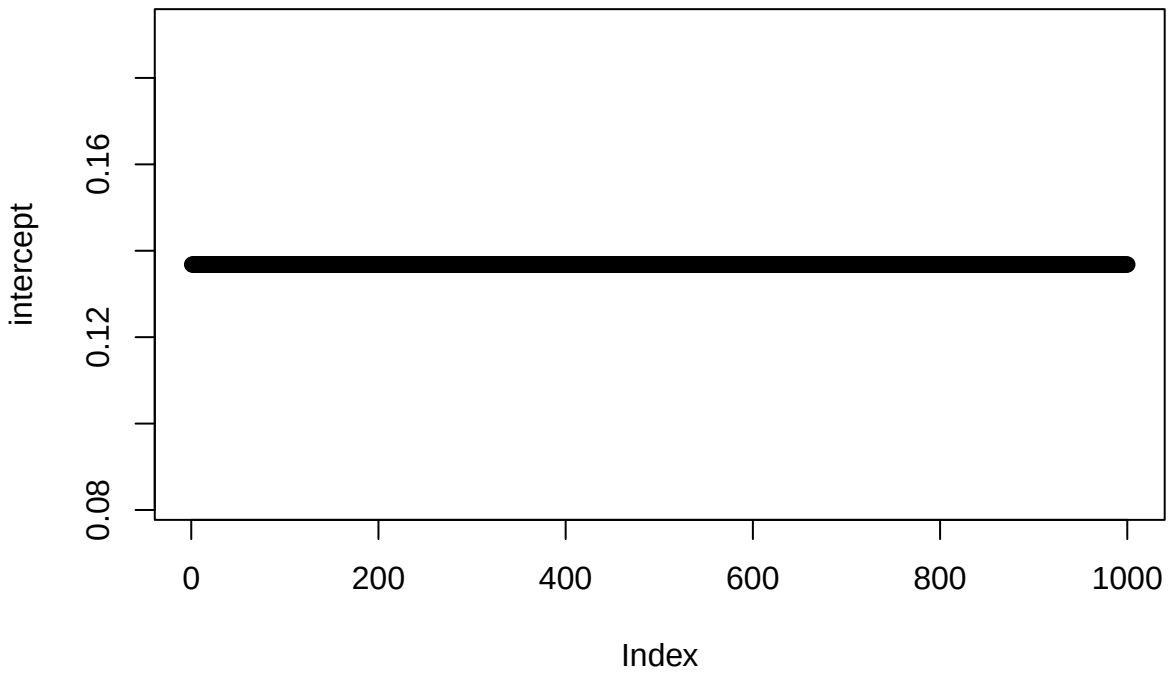
Perform some simulations to compare the length and confidence level of the studentized bootstrap confidence interval (from Problem 1) and of the student confidence interval (the classical one). Compare them at various sample sizes and in settings involving different distributions, for example, the normal distribution and a skewed distribution like the exponential distribution (centered to have mean 0). In the code, first briefly explain in words what you intend to do, and then do it, and at the end offer some brief comments on the results of your simulation study.

##To do: 1) Perform a simulation to run function bootLS() 1000 times and infer the length of confidence level per iteration and draw conclusion. We will be doing this after setting the seed, as done in lab. If we dont

set the seed, we will get different lengths every iteration. 2) Create a data with with normal distribution, (sigma = 0.5) and do the simulation to compare them at various sample sizes and different confidence level. Draw conclusions. 3) Repeat the above experiment but this time in settings involving a skewed distribution, here exponential distribution (centered to have mean 0). Draw conclusions. 4) Draw the final conclusion.

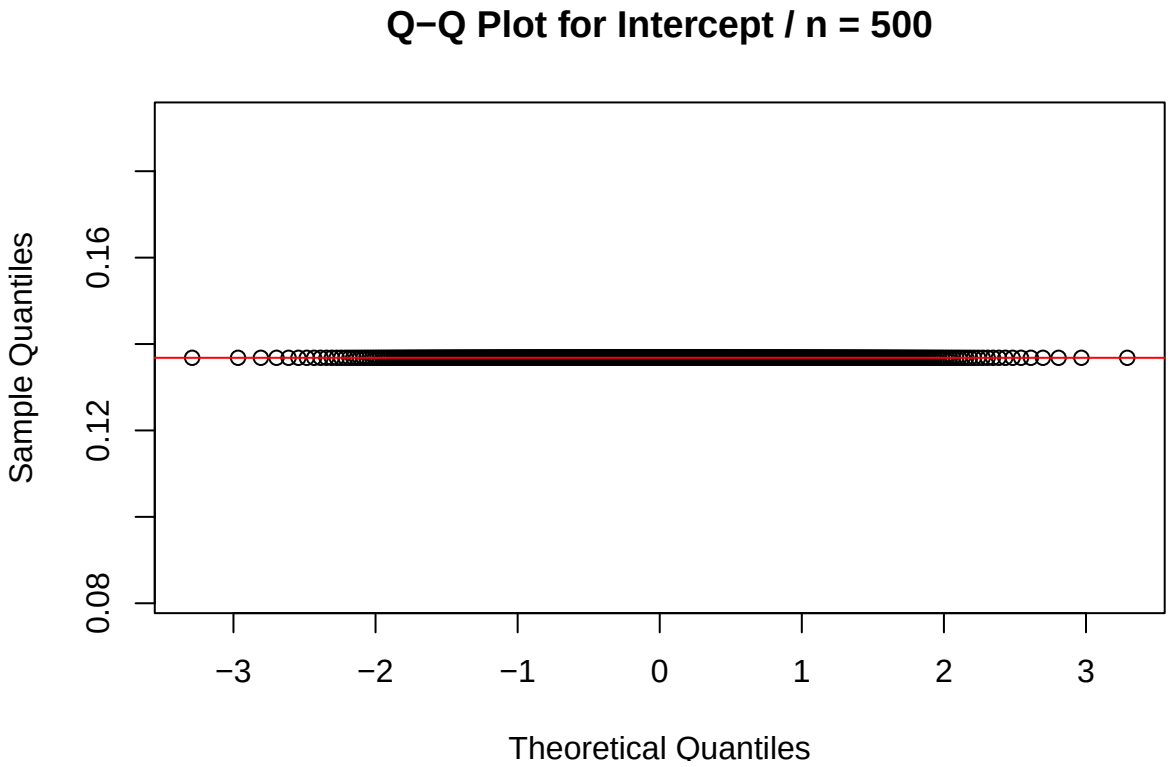
##Note: In the interest of time we have used n_simulation = 250, instead of 1000, the results aren't affected by this change.

```
#' ## Part A - Showing that the length of confidence interval remains same over 1000 iterations
N = 1000
intercept = numeric(N)
slope = numeric(N)
n = 500                  # modify the sample size to be {50, 100, 200, 500}
set.seed(2022)
x = runif(n, -1, 1)
err = rnorm(n, 0, 0.5)         # normal distribution
#err = rexp(n, rate = 0.5) # Skewed distribution, exponential
y = 1 + 2 * x + err

#Looping
for (j in 1:N){
  fit = lm(y ~ x)
  interval= bootLS(x,y, B=n)
  intercept[j] = interval[4]-interval[3]
  slope[j] = interval[1]-interval[2]
}

plot(intercept)
```



##Inference: If we set the seed then the standardized quartiles comes out to same every time. Which is expected.

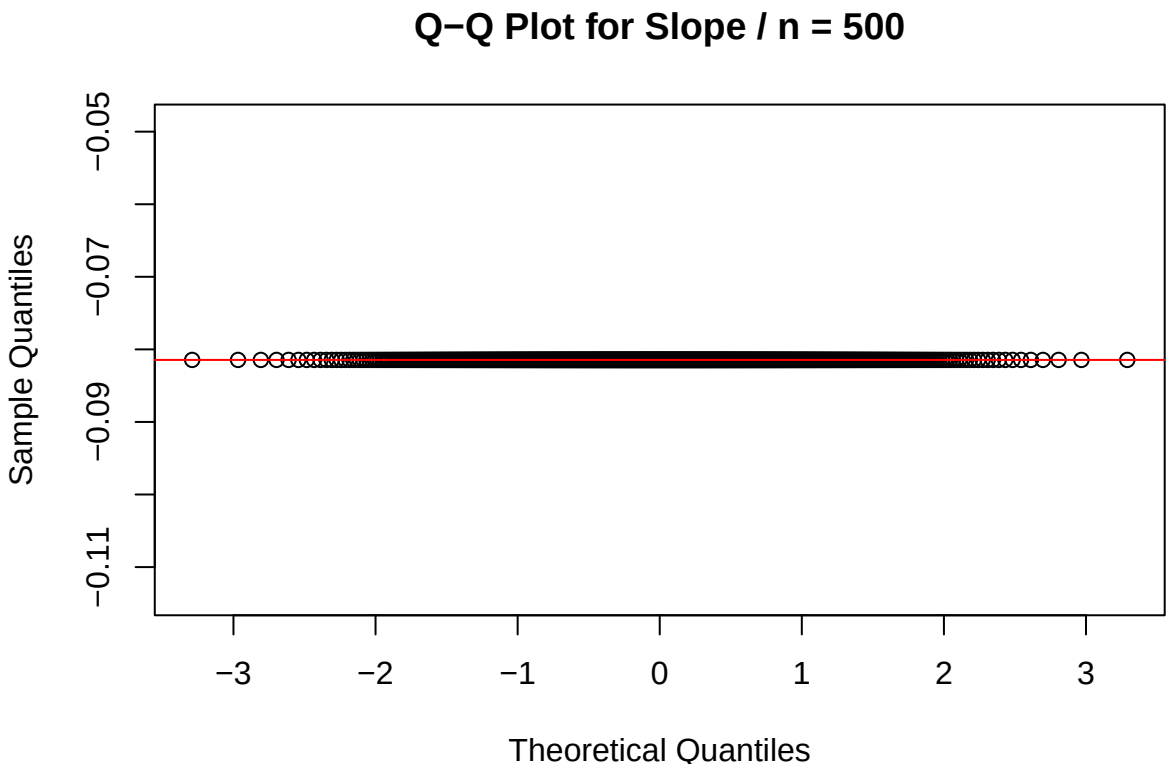
```
#' Q-Q plot
qqnorm(intercept, main = paste("Q-Q Plot for Intercept / n =", n))
qqline(intercept, col = "red")
```



##In-

ference: We see that the intercepts for the bootstrap comes out to be constant.

```
#' Q-Q plot for the slope to check for marginal normality
qqnorm(slope, main = paste("Q-Q Plot for Slope / n =", n))
qqline(slope, col = "red")
```



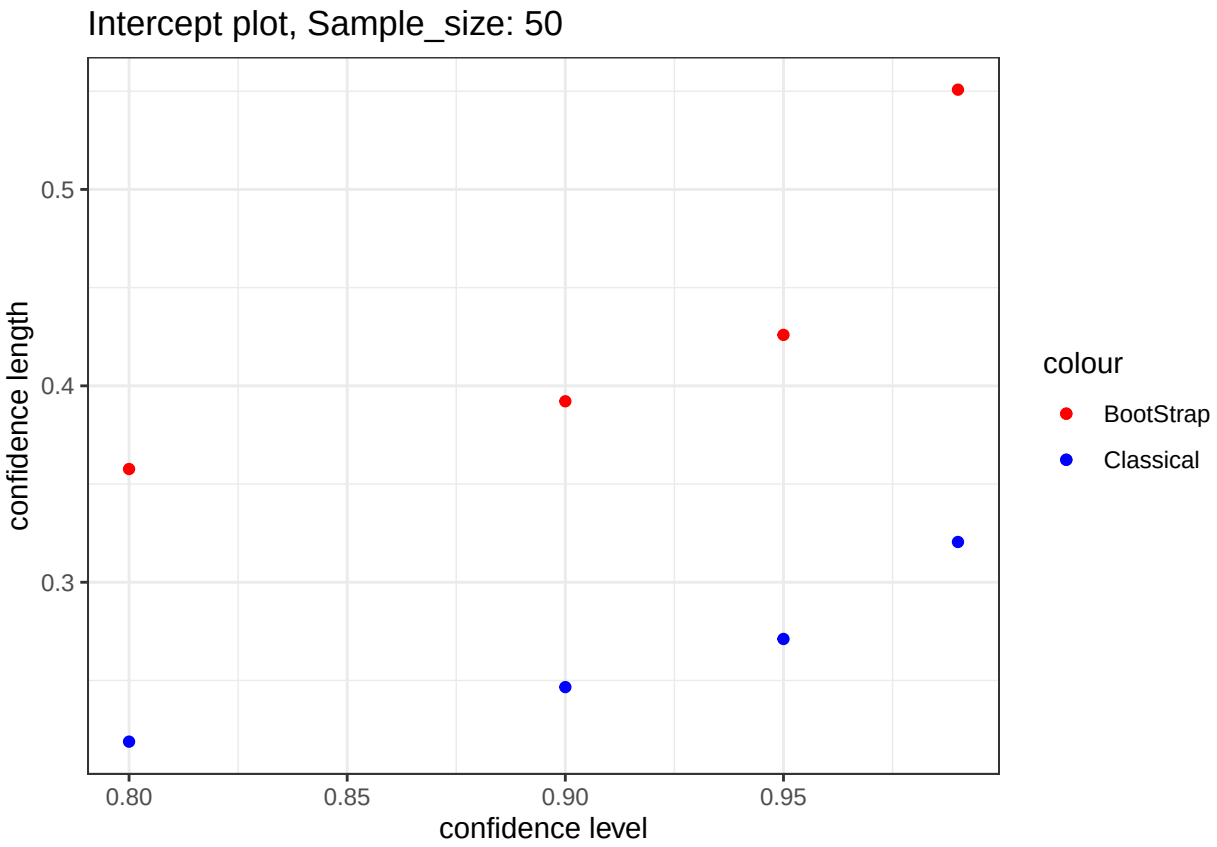
##In-

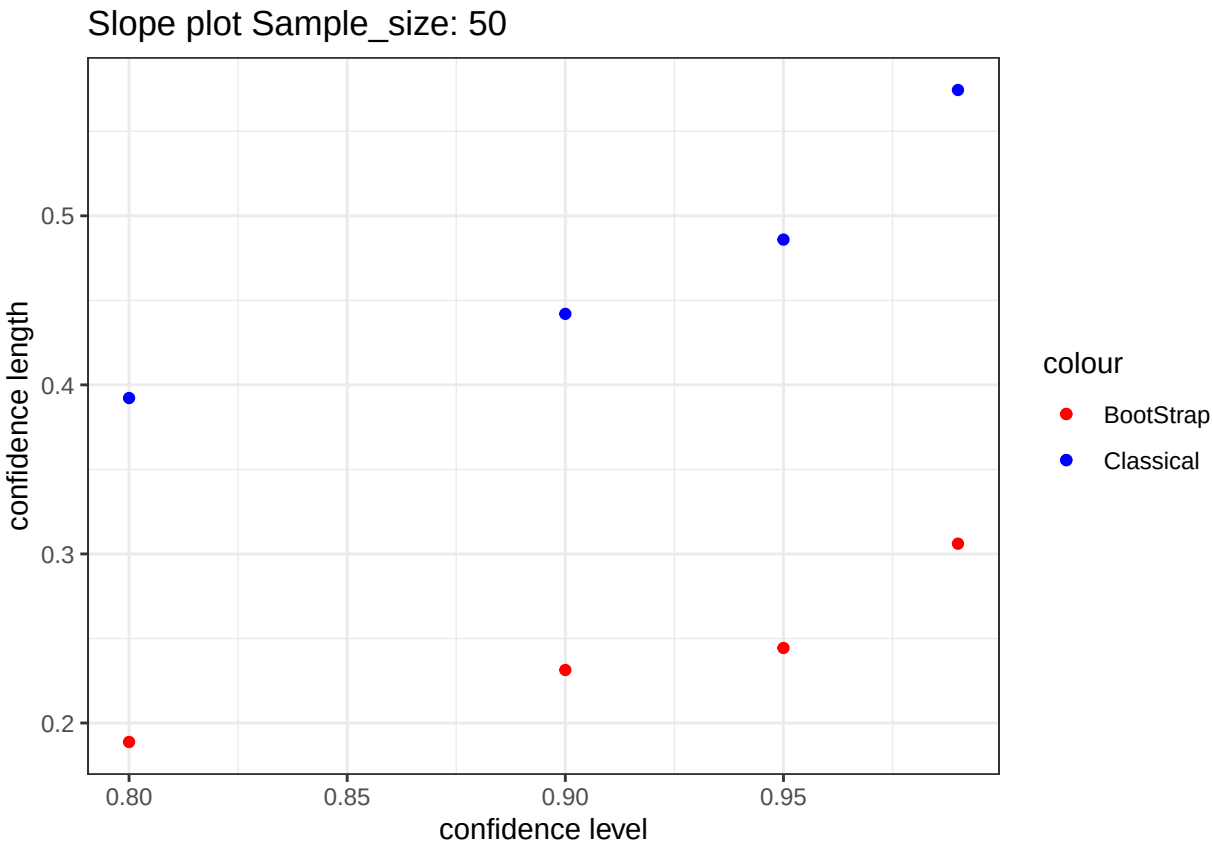
ference: We see that the slope for the bootstrap comes out to be constant.

```r
sample_size = c(50,100,500)
conf_level = c(0.8,0.9, 0.95, 0.99)
N_sim = 10

for (n in sample_size){
  x = runif(n, -1, 1)
  n_err = rnorm(n, 0, 0.5)
  #e_err = rexp(n, rate = 0.5)
  #e_err = e_err - mean(e_err)
  y = 1 + 2 * x + n_err

  intercept_len_boot = rep(NA,4)
  slope_len_boot = rep(NA,4)
  intercept_len_classic = rep(NA,4)
  slope_len_classic = rep(NA,4)
  intercept_per_sim = rep(NA,N_sim)
  slope_per_sim = rep(NA,N_sim)


  print(n)
  for (i in 1:4){
    conf = conf_level[i]
    print(conf)
    fit = lm(y ~ x)

    for (j in 1:N_sim){
      interval= bootLS(x,y,conf = conf, B=n)
      intercept_per_sim[j] = interval[4]-interval[3]
      slope_per_sim[j] = interval[2]-interval[1]
    }
    intercept_len_boot[i] = mean(intercept_per_sim)
    slope_len_boot[i] = mean(slope_per_sim)
    intercept_len_classic[i] = confint(fit, level = conf)[1,][2]-confint(fit, level = 0.9)[1,][1]
    slope_len_classic[i] = confint(fit, level = conf)[2,][2]-confint(fit, level = 0.9)[2,][1]
  }


  #print(intercept_len_boot)
  df_1 <- data.frame(x = c(0.8,0.9,0.95,0.99),
                     y1 = intercept_len_boot, y2 = intercept_len_classic)
  df_2 <- data.frame(x = c(0.8,0.9,0.95,0.99),
                     y1 = slope_len_boot, y2 = slope_len_classic)

  print(ggplot(df_1, aes(x)) +
      geom_point(aes(y = y1, color = "BootStrap")) +
      geom_point(aes(y = y2, color = "Classical")) +
      scale_color_manual(values = c("BootStrap" = "red", "Classical" = "blue")) +
      labs(title = paste("Intercept plot, Sample_size:", n),
           x = "confidence level", y = "confidence length") +
      theme_bw())


  print(ggplot(df_2, aes(x)) +
```

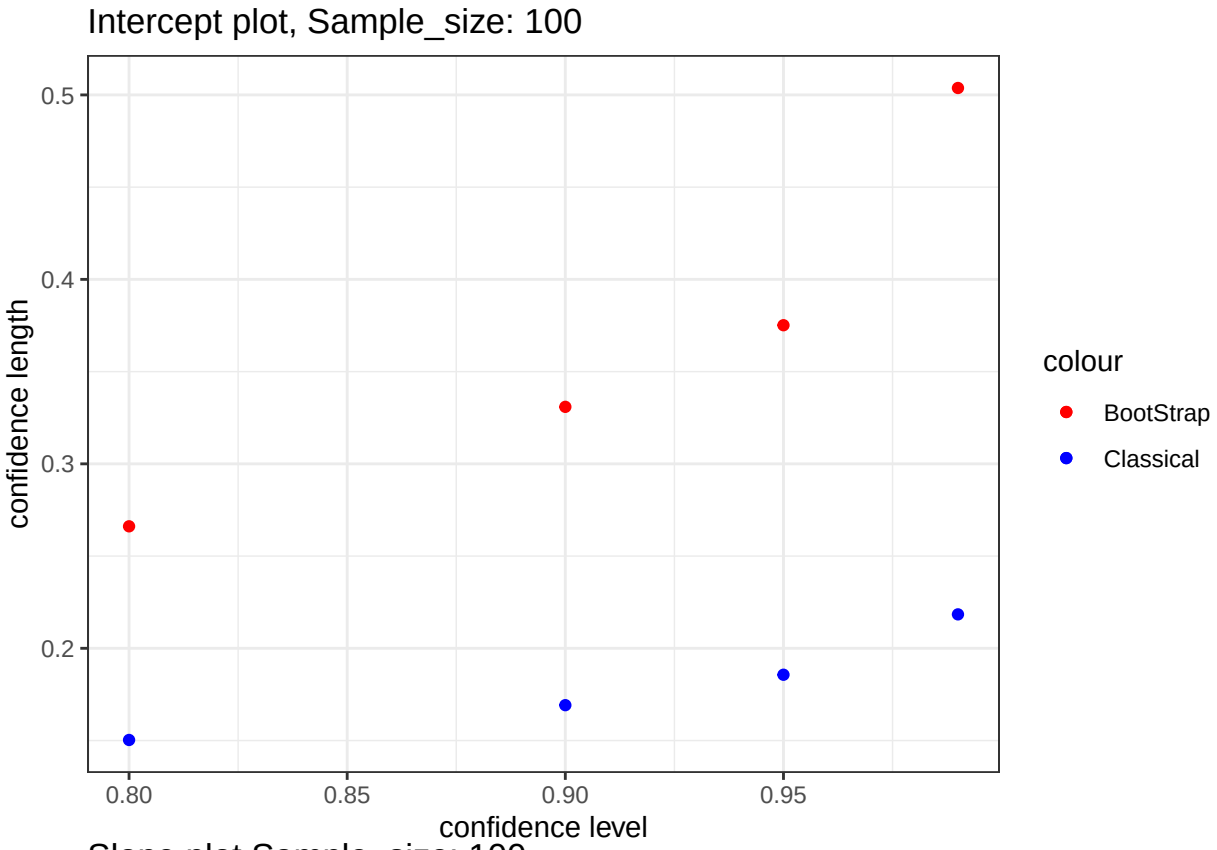
```
      geom_point(aes(y = y1, color = "BootStrap")) +
      geom_point(aes(y = y2, color = "Classical")) +
      scale_color_manual(values = c("BootStrap" = "red", "Classical" = "blue")) +
      labs(title = paste("Slope plot Sample_size:", n),
           x = "confidence level", y = "confidence length") +
      theme_bw())

}
```

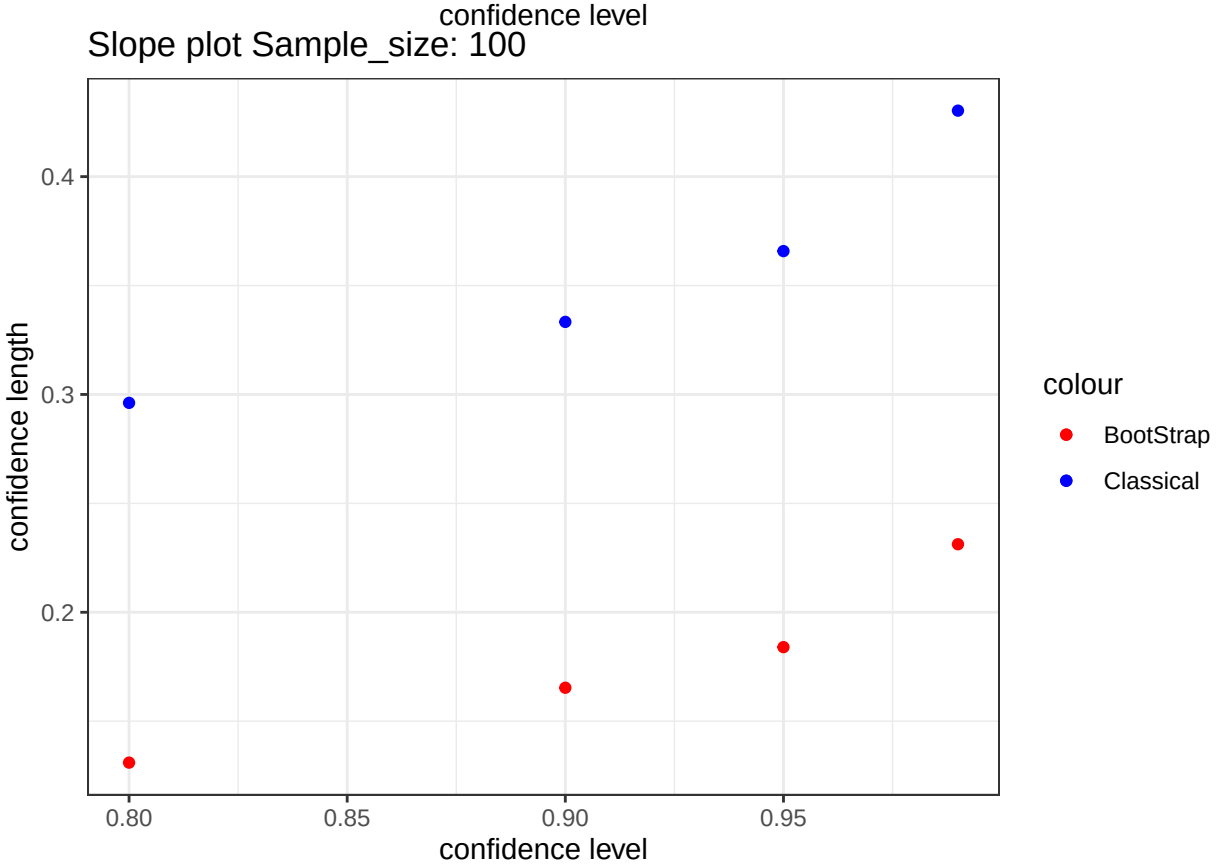
```
## [1] 50
## [1] 0.8
## [1] 0.9
## [1] 0.95
## [1] 0.99
```





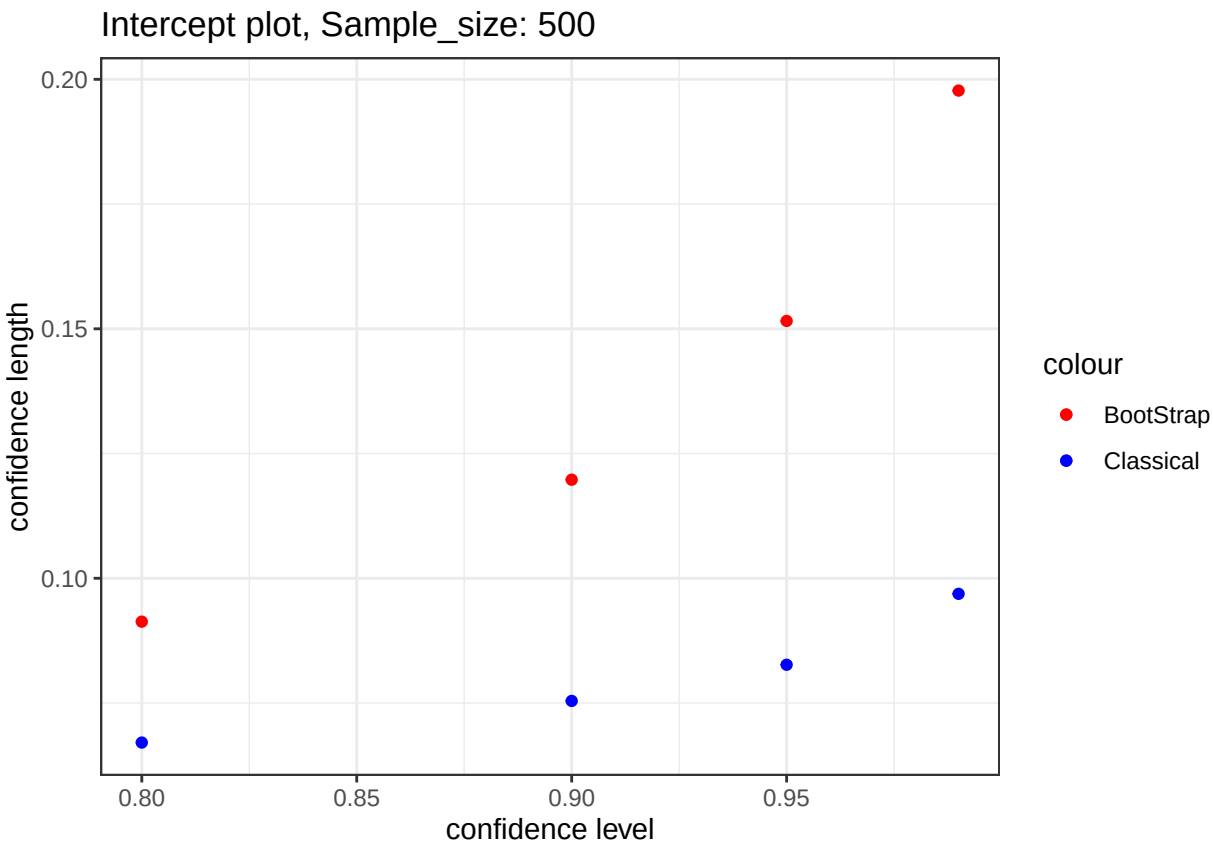

```
## [1] 100
## [1] 0.8
## [1] 0.9
## [1] 0.95
## [1] 0.99
```

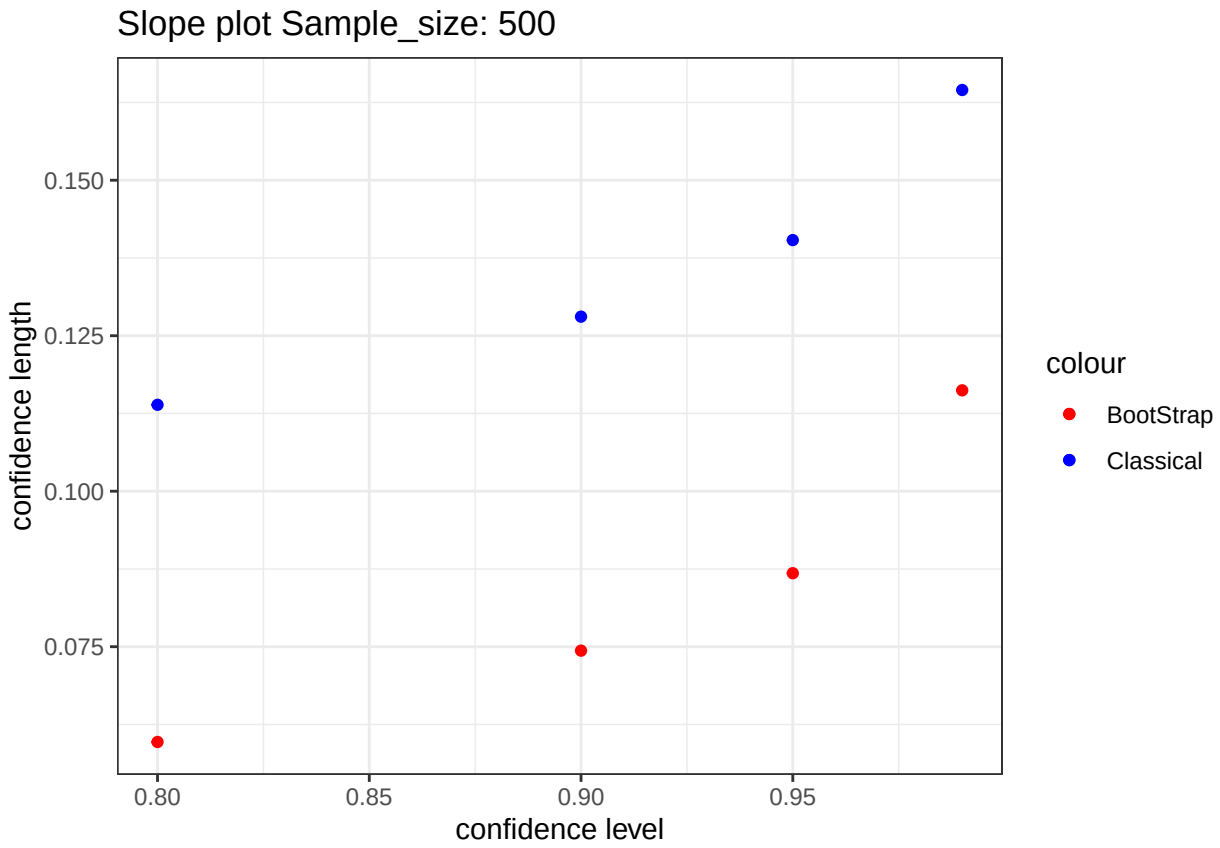

Intercept plot, Sample_size: 100
confidence length
0.5
0.4
0.3
0.2
0.80
0.85
0.90
0.95
confidence level
colour
BootStrap
Classical


Slope plot Sample_size: 100
confidence length
0.4
0.3
0.2
0.80
0.85
0.90
0.95
confidence level
colour
BootStrap
Classical

```
## [1] 500
```

```
## [1] 0.8
## [1] 0.9
## [1] 0.95
## [1] 0.99
```





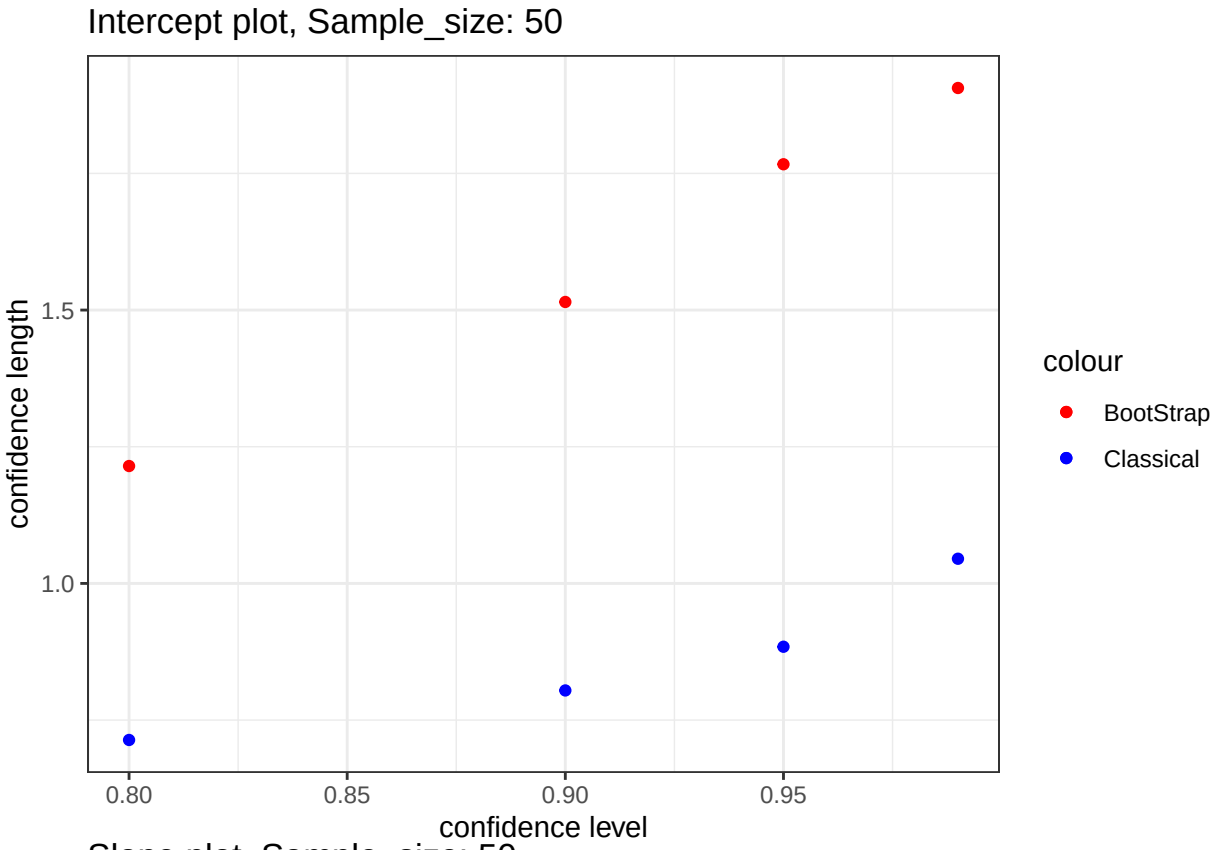
##Inference: In the graphs we can see that: 1) As the confidence level increases the length of confidence interval increases. 2) The length of studentized bootstrap confidence interval is more than that of the student confidence interval (the classical one). Since it is more reliable and accurate method of estimating confidence intervals, especially when the assumptions of normality are violated or the sample size is small. 3) As the sample size increases the length of confidence interval decreases.

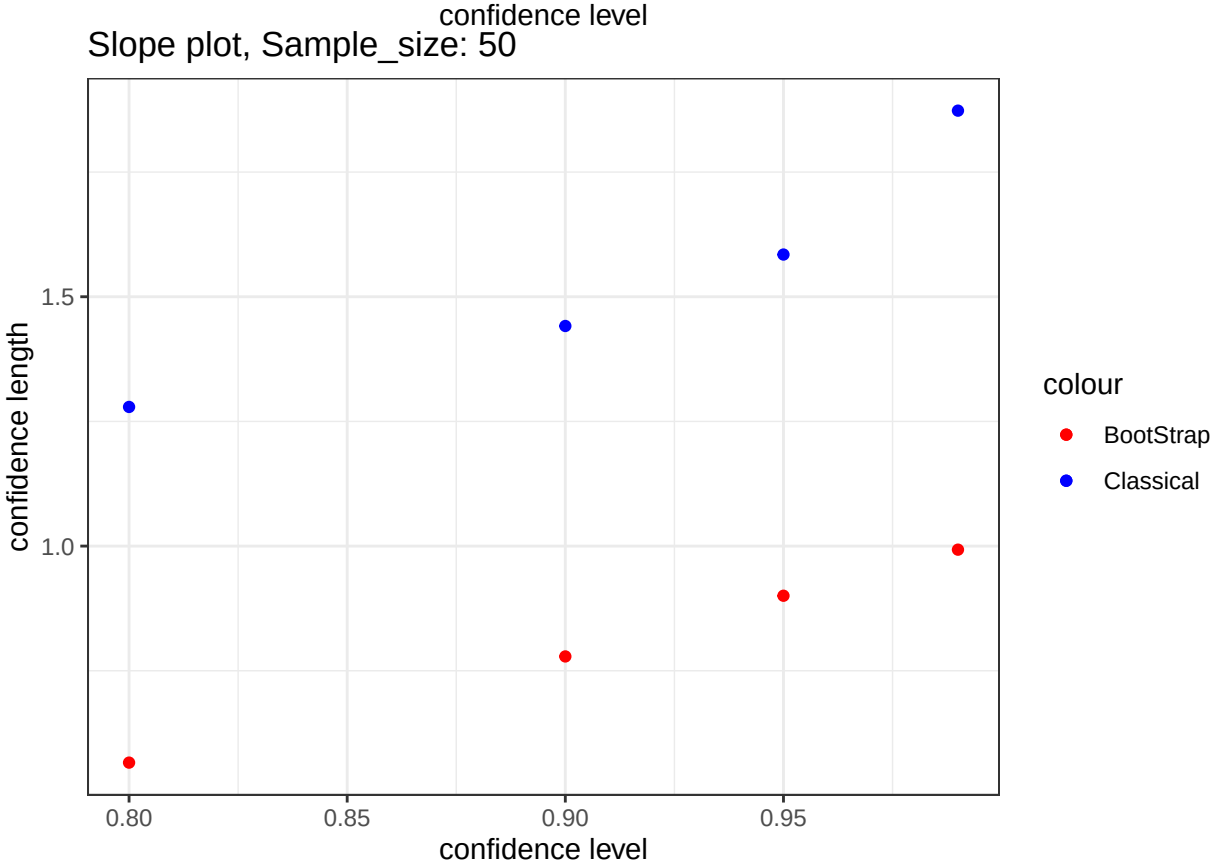
Exponential distribution

```
for (n in sample_size){
  x = runif(n, -1, 1)
  #n_err = rnorm(n, 0, 0.5)
  e_err = rexp(n, rate = 0.5)
  e_err = e_err - mean(e_err)
  y = 1 + 2 * x + e_err

  intercept_len_boot = rep(NA,4)
  slope_len_boot = rep(NA,4)
  intercept_len_classic = rep(NA,4)
  slope_len_classic = rep(NA,4)
  intercept_per_sim = rep(NA,N_sim)
  slope_per_sim = rep(NA,N_sim)


  print(n)
  for (i in 1:4){
    conf = conf_level[i]
    print(conf)
    fit = lm(y ~ x)
```

```
    for (j in 1:N_sim){
      interval= bootLS(x,y,conf = conf, B=n)
      intercept_per_sim[j] = interval[4]-interval[3]
      slope_per_sim[j] = interval[2]-interval[1]
    }
    intercept_len_boot[i] = mean(intercept_per_sim)
    slope_len_boot[i] = mean(slope_per_sim)
    intercept_len_classic[i] = confint(fit, level = conf)[1,][2]-confint(fit, level = 0.9)[1,][1]
    slope_len_classic[i] = confint(fit, level = conf)[2,][2]-confint(fit, level = 0.9)[2,][1]
  }


  #print(intercept_len_boot)
  df_1 <- data.frame(x = c(0.8,0.9,0.95,0.99),
                     y1 = intercept_len_boot, y2 = intercept_len_classic)
  df_2 <- data.frame(x = c(0.8,0.9,0.95,0.99),
                     y1 = slope_len_boot, y2 = slope_len_classic)

  print(ggplot(df_1, aes(x)) +
      geom_point(aes(y = y1, color = "BootStrap")) +
      geom_point(aes(y = y2, color = "Classical")) +
      scale_color_manual(values = c("BootStrap" = "red", "Classical" = "blue")) +
      labs(title = paste("Intercept plot, Sample_size:", n),
             x = "confidence level", y = "confidence length") +
      theme_bw())


  print(ggplot(df_2, aes(x)) +
      geom_point(aes(y = y1, color = "BootStrap")) +
      geom_point(aes(y = y2, color = "Classical")) +
      scale_color_manual(values = c("BootStrap" = "red", "Classical" = "blue")) +
      labs(title = paste("Slope plot, Sample_size:", n),
                         x = "confidence level", y = "confidence length") +
      theme_bw())

}
```

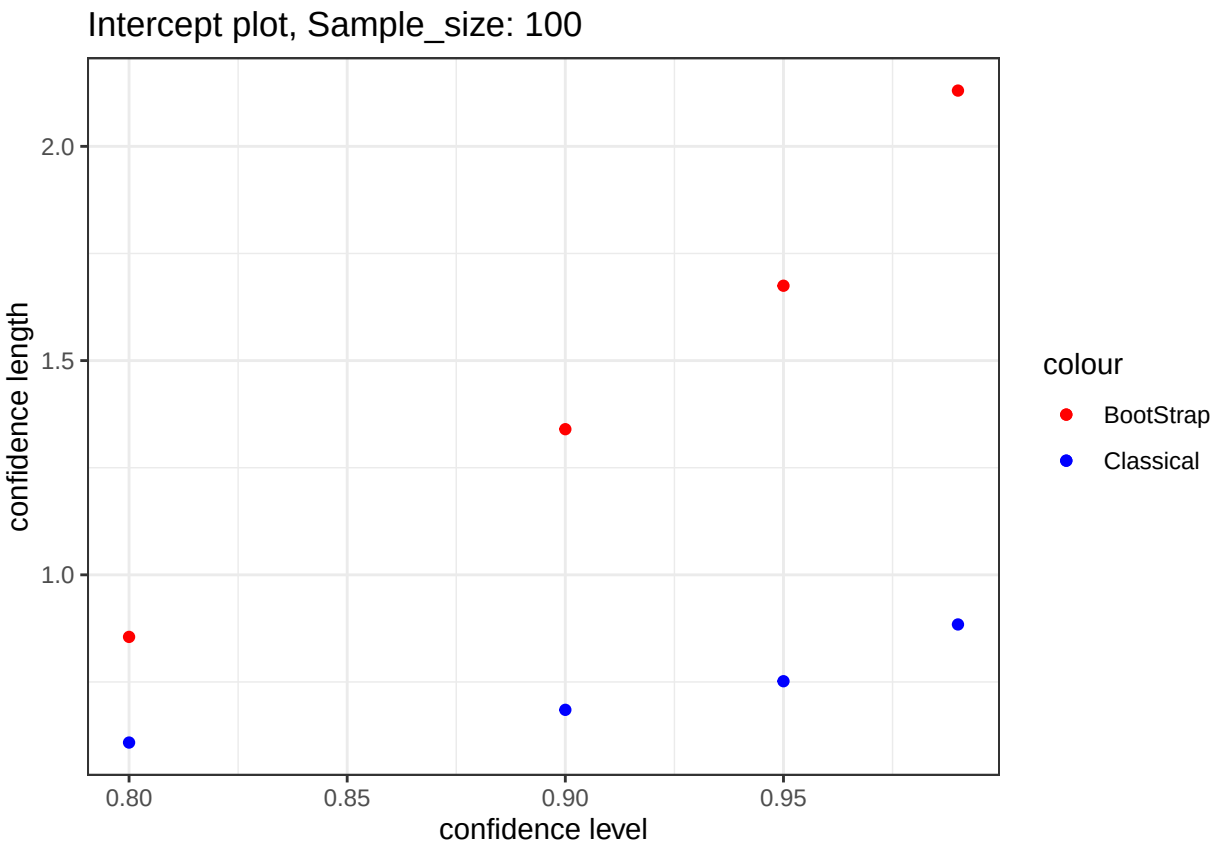
```
## [1] 50
## [1] 0.8
## [1] 0.9
## [1] 0.95
## [1] 0.99
```

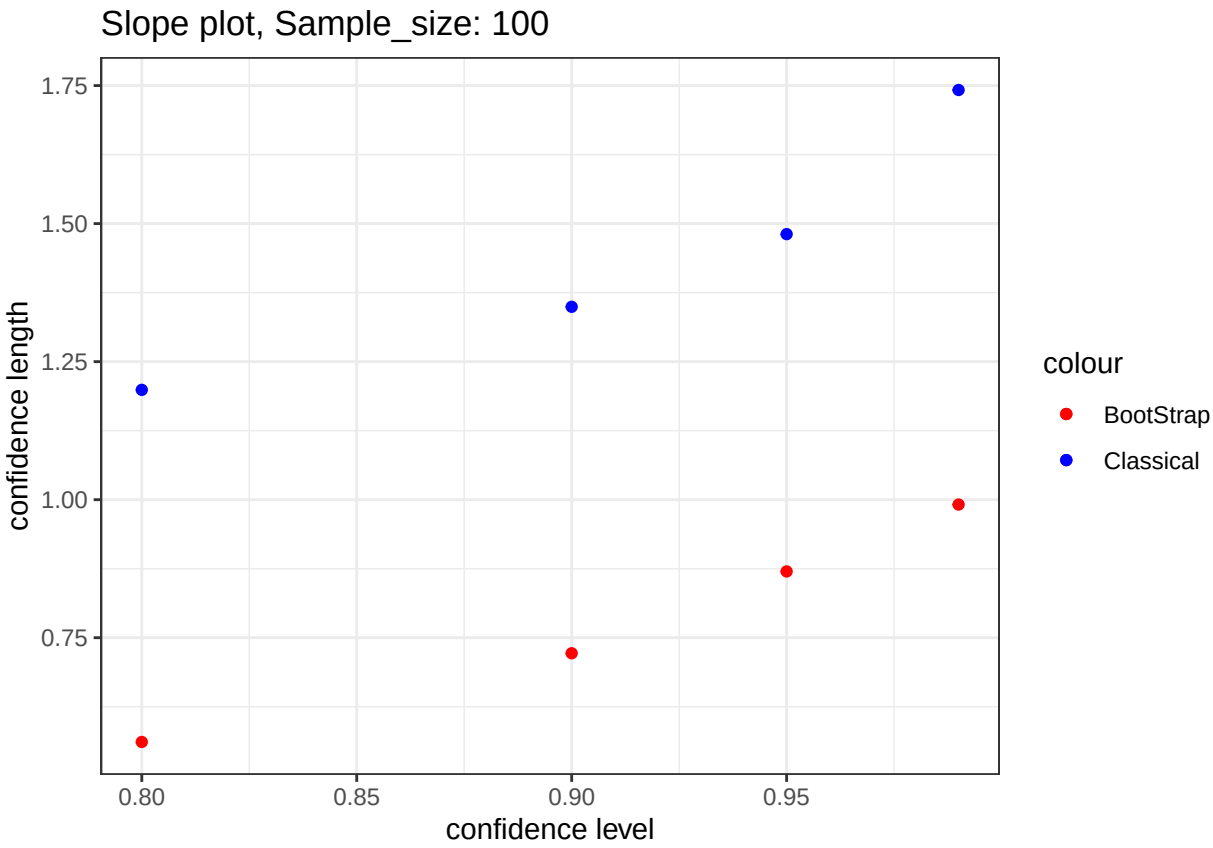

Intercept plot, Sample_size: 50
confidence length
1.5
1.0
0.80
0.85
0.90
0.95
confidence level
colour
BootStrap
Classical


Slope plot, Sample_size: 50
confidence length
1.5
1.0
0.80
0.85
0.90
0.95
confidence level
colour
BootStrap
Classical

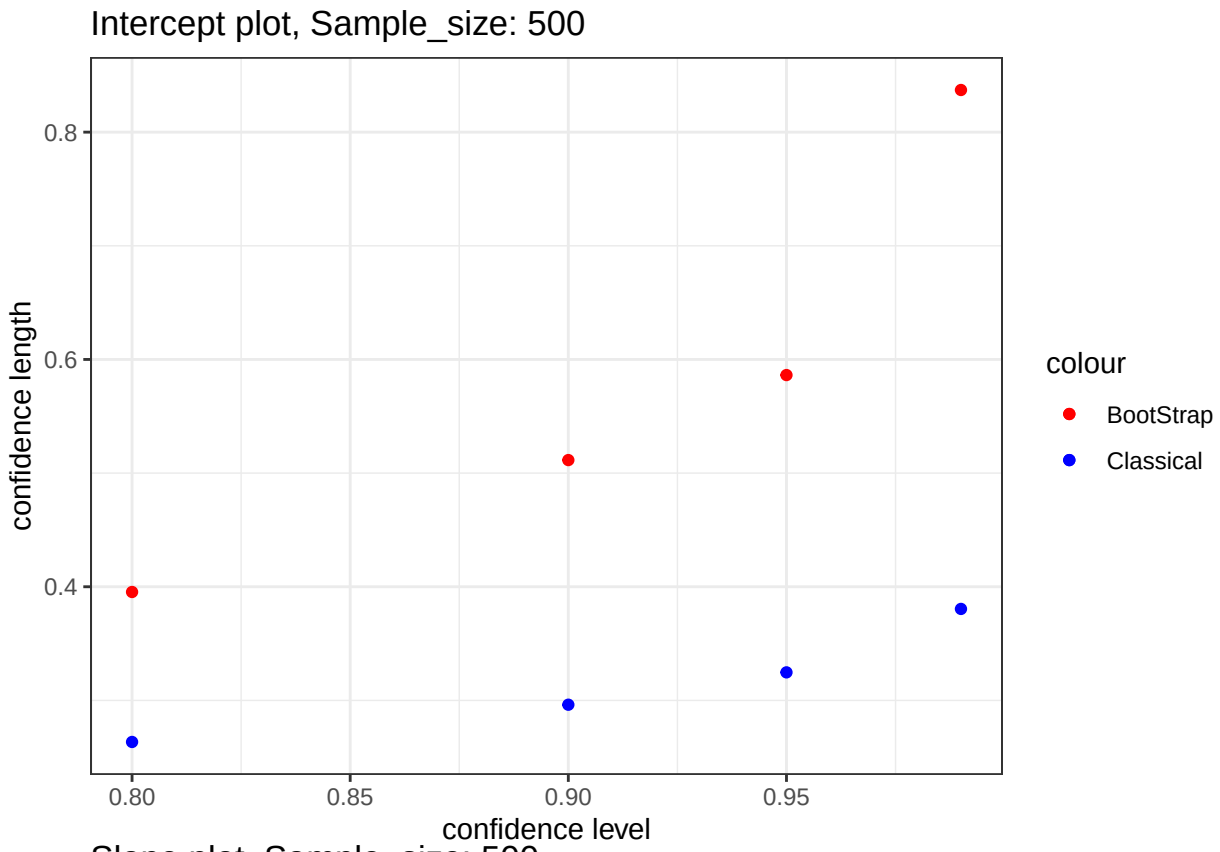
```
## [1] 100
```

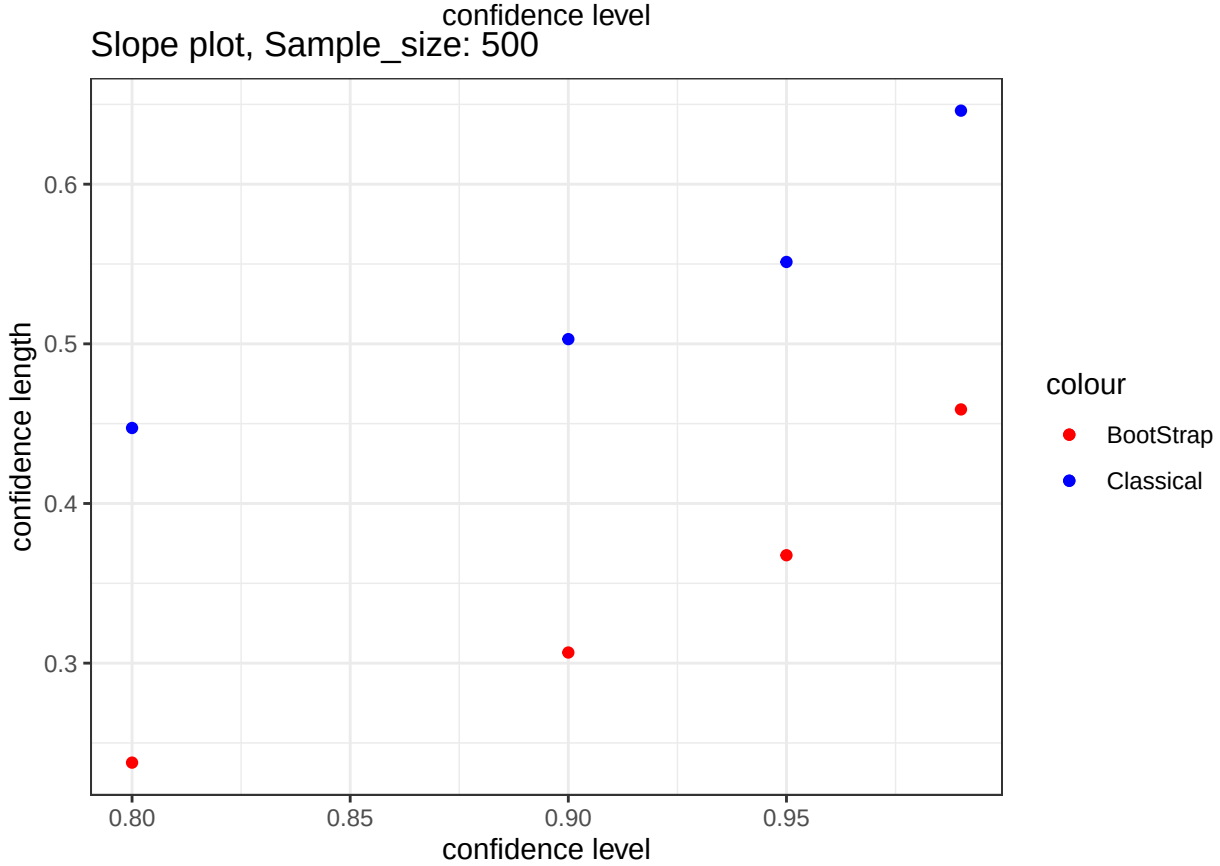
```
## [1] 0.8
## [1] 0.9
## [1] 0.95
## [1] 0.99
```






```
## [1] 500
## [1] 0.8
## [1] 0.9
## [1] 0.95
## [1] 0.99
```





##Inference: 1) When we apply skewed distribution, here exponential, then the length of the confidence

interval increases when compared to corresponding length of confidence interval for normal distributed error.The other trends are same as the normal distributed data, i.e. 2) As the confidence level increases the length of confidence interval increases. 3) The length of studentized bootstrap confidence interval is more than that of the student confidence interval (the classical one). Since it is more reliable and accurate method of estimating confidence intervals, especially when the assumptions of normality are violated or the sample size is small.
4) As the sample size increases the length of confidence interval decreases.

##Final Inference: The normal confidence interval assumes that the sampling distribution of the sample mean (or other statistic of interest) is normal, which is often not the case in real-world situations. Additionally, the normal confidence interval can be biased or too narrow in small samples or when the population distribution is not normal.

On the other hand, studentized bootstrap confidence intervals are calculated based on resampling the observed data and constructing a sampling distribution of the sample statistic of interest. This method does not make any assumptions about the distribution of the population, and it can provide a more accurate estimate of the confidence interval, especially when the sample size is small or the population distribution is unknown.

Therefore, the inference that can be taken from the comparison of these two methods is that studentized bootstrap confidence intervals can be a more reliable and accurate method of estimating confidence intervals, especially when the assumptions of normality are violated or the sample size is small. However, it should be noted that bootstrap methods can be computationally intensive and may take longer to compute than normal-based methods.

##Team Contributions: Both the team members Sourabh Prakash and Priyanshi Shah have contributed equally to the homework by discussing the key points and logic together and doing pair programming. For the implementation part question 1 was contributed by Priyanshi Shah and question 2 by Sourabh Prakash. The inferences were drawn together by both the team members.