# HW 6 - Due Tuesday October 18, 2016 in moodle and hardcopy in class.

# Upload R file to Moodle with name: HW6\_490IDS\_YourClassID.R

# Do not remove any of the comments. These are marked by #

# Please ensure that no identifying information (other than your class ID)

# is on your paper copy, including your name

# We will use the bootstrap technique to generate confidence intervals

# 1. Suppose we have a sample of data from an exponential distribution

# with parameter lambda. In this case use lambda.hat = 1/mean(X).

# As the number of observations increases, does the estimate for lambda

# become roughly normally distributed? We will answer this question in

# the following parts.

# 1a. (1) Generate 100 observations of test data, with lambda=3. Remember

# to set your seed before carrying out any computations.

set.seed(0)

test = rexp(100, rate = 3)

# 1b. (1) What is the mean of your test data? (give the code and the value)

mean(test)

# The mean is 0.344049.

# 1c. (1) What is your estimate lambda.hat? (give the code and the value)

lambda.hat = 1 / mean(test)

# My estimate of lambda.hat is 2.906563.

# 2. Now use the bootstrap to estimate the distribution of

# lambda.hat and create bootstrap confidence intervals for lambda,

# rather than the approach in 1).

# 2a. (1) Form a set of bootstrap estimates of our parameter by generating B

# random samples as you did once in 1a but use lambda.hat since we do not

# know the true lambda in this case (keep n=100). Set B=1000, and again set

# your seed.

> set.seed(0)

> B = 1000

> boot.samples = matrix(nrow = B, ncol = 100)

> for (i in 1:B) {

+ boot.samples[i,] = rexp(100, rate = lambda.hat)

+ }

# 2b. (1) Get a new estimate for lambda.hat from each of the bootstrap samples

# in 2a. You'll want to create a matrix to receive each value. You should

# have 1000 estimates for lambda.hat now.

> boot.samples.estimates = matrix(nrow = 1, ncol = B)

> for (i in 1:B) {

+ boot.samples.estimates[i] = 1/mean(boot.samples[i,])

+ }

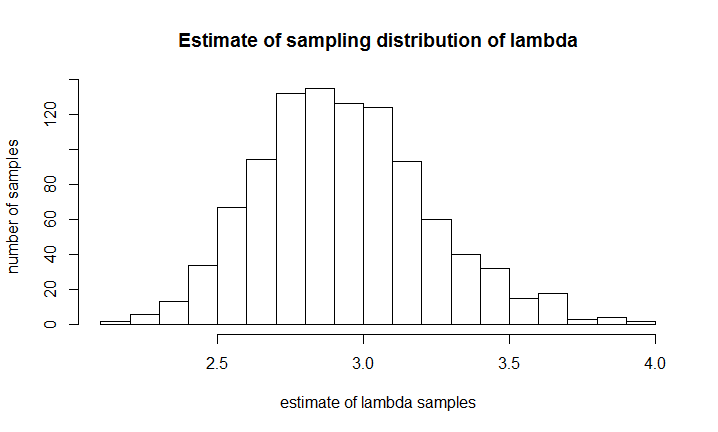
# 2c. (2) Now look at the sampling distribution for lambda.hat, using the hist

# function. Remember the graphing techniques discussed in class and use them

# to make the plot look professional. Does the distribution look normal?

hist(boot.samples.estimates, main="Estimate of sampling distribution of lambda", xlab = "estimate of lambda samples", ylab = "number of samples", breaks = 20)

# The distribution looks normal.



# 2d. (1) Calculate an estimate of the standard error of lambda.hat using your

# collection of bootstrap estimated parameters. What is your 95% confidence interval?

# I use Z=2 here because it is not perfectly normal distribution.

# If we only keep two digits after the decimal point, there will hardly be any difference between Z=2 and Z=1.96.

> SE = sd(boot.samples.estimates)

> CI = c(mean(boot.samples.estimates)-2\*SE, mean(boot.samples.estimates)+2\*SE)

# The 95% confidence interval is (2.34,3.53).

# 3a. (5) We made some decisions when we used the bootstrap above that we can now question.

# Repeat the above creation of a confidence interval for a range of values of data

# (we had our sample size fixed at 100) and a range of bootstrap values (we had B

# fixed at 1000). Suppose the sample size varies (100, 200, 300, .... , 1000) and

# B varies (1000, 2000, ... , 10000). You will likely find it useful to write

# functions to carry out these calculations. Your final output should be

# upper and lower pairs for the confidence intervals produced using the bootstrap

# method for each value of sample size and B.

# generalize 2b into a function, and vary inputs of sample size and B as we did above.

boot.sample <- function(sample.size, B){

#code here

set.seed(0)

test = rexp(100, rate = 3)

lambda.hat = 1 / mean(test)

boot.estimates = matrix(nrow = 1, ncol = B)

for (i in 1:B) {

boot.estimates[,i] = 1/mean(rexp(sample.size,lambda.hat))

}

SE = sd(boot.estimates)

upperCI = mean(boot.estimates)+2\*SE

lowerCI = mean(boot.estimates)-2\*SE

return(c(lowerCI,mean(boot.estimates),upperCI))

}

CI100sample = matrix(nrow = 3, ncol = 10)

for (i in 1:10) {

CI100sample[1,i] = boot.sample(sample.size = 100, B = i\*1000)[1]

CI100sample[2,i] = boot.sample(sample.size = 100, B = i\*1000)[2]

CI100sample[3,i] = boot.sample(sample.size = 100, B = i\*1000)[3]

}

CI200sample = matrix(nrow = 3, ncol = 10)

for (i in 1:10) {

CI200sample[1,i] = boot.sample(sample.size = 200, B = i\*1000)[1]

CI200sample[2,i] = boot.sample(sample.size = 200, B = i\*1000)[2]

CI200sample[3,i] = boot.sample(sample.size = 200, B = i\*1000)[3]

}

CI300sample = matrix(nrow = 3, ncol = 10)

for (i in 1:10) {

CI300sample[1,i] = boot.sample(sample.size = 300, B = i\*1000)[1]

CI300sample[2,i] = boot.sample(sample.size = 300, B = i\*1000)[2]

CI300sample[3,i] = boot.sample(sample.size = 300, B = i\*1000)[3]

}

CI400sample = matrix(nrow = 3, ncol = 10)

for (i in 1:10) {

CI400sample[1,i] = boot.sample(sample.size = 400, B = i\*1000)[1]

CI400sample[2,i] = boot.sample(sample.size = 400, B = i\*1000)[2]

CI400sample[3,i] = boot.sample(sample.size = 400, B = i\*1000)[3]

}

CI500sample = matrix(nrow = 3, ncol = 10)

for (i in 1:10) {

CI500sample[1,i] = boot.sample(sample.size = 500, B = i\*1000)[1]

CI500sample[2,i] = boot.sample(sample.size = 500, B = i\*1000)[2]

CI500sample[3,i] = boot.sample(sample.size = 500, B = i\*1000)[3]

}

CI600sample = matrix(nrow = 3, ncol = 10)

for (i in 1:10) {

CI600sample[1,i] = boot.sample(sample.size = 600, B = i\*1000)[1]

CI600sample[2,i] = boot.sample(sample.size = 600, B = i\*1000)[2]

CI600sample[3,i] = boot.sample(sample.size = 600, B = i\*1000)[3]

}

CI700sample = matrix(nrow = 3, ncol = 10)

for (i in 1:10) {

CI700sample[1,i] = boot.sample(sample.size = 700, B = i\*1000)[1]

CI700sample[2,i] = boot.sample(sample.size = 700, B = i\*1000)[2]

CI700sample[3,i] = boot.sample(sample.size = 700, B = i\*1000)[3]

}

CI800sample = matrix(nrow = 3, ncol = 10)

for (i in 1:10) {

CI800sample[1,i] = boot.sample(sample.size = 800, B = i\*1000)[1]

CI800sample[2,i] = boot.sample(sample.size = 800, B = i\*1000)[2]

CI800sample[3,i] = boot.sample(sample.size = 800, B = i\*1000)[3]

}

CI900sample = matrix(nrow = 3, ncol = 10)

for (i in 1:10) {

CI900sample[1,i] = boot.sample(sample.size = 900, B = i\*1000)[1]

CI900sample[2,i] = boot.sample(sample.size = 900, B = i\*1000)[2]

CI900sample[3,i] = boot.sample(sample.size = 900, B = i\*1000)[3]

}

CI1000sample = matrix(nrow = 3, ncol = 10)

for (i in 1:10) {

CI1000sample[1,i] = boot.sample(sample.size = 1000, B = i\*1000)[1]

CI1000sample[2,i] = boot.sample(sample.size = 1000, B = i\*1000)[2]

CI1000sample[3,i] = boot.sample(sample.size = 1000, B = i\*1000)[3]

}

# 3b. (2) Plot your CI limits to show the effect of changing the sample size and

# changing the number of bootstrap replications. What do you conclude?

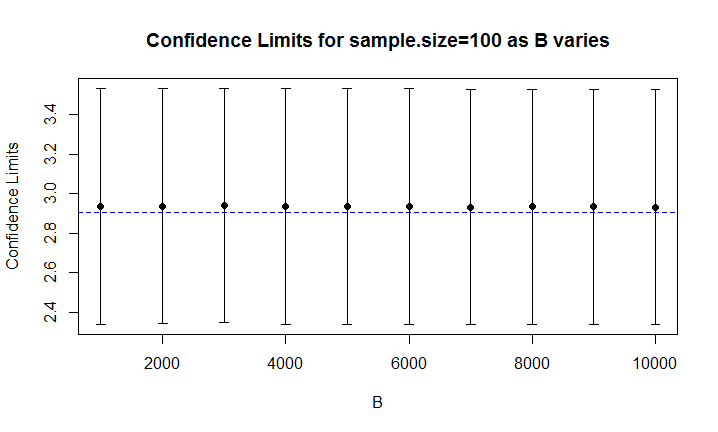
install.packages("Hmisc")

library("Hmisc", lib.loc="C:/Program Files for operation/R-3.3.1/library")

errbar(x=seq(1000,10000,by=1000),y=CI100sample[2,],yplus=CI100sample[3,],yminus=CI100sample[1,],main = "Confidence Limits for sample.size=100 as B varies",xlab = "B",ylab = "Confidence Limits")

title(main = "Confidence Limits for sample.size=100 as B varies")

abline(h=lambda.hat,col="blue",lty=2)



errbar(x=seq(100,1000,by=100),y=c(CI100sample[2,1],CI200sample[2,1],CI300sample[2,1],CI400sample[2,1],CI500sample[2,1],CI600sample[2,1],CI700sample[2,1],CI800sample[2,1],CI900sample[2,1],CI1000sample[2,1]),

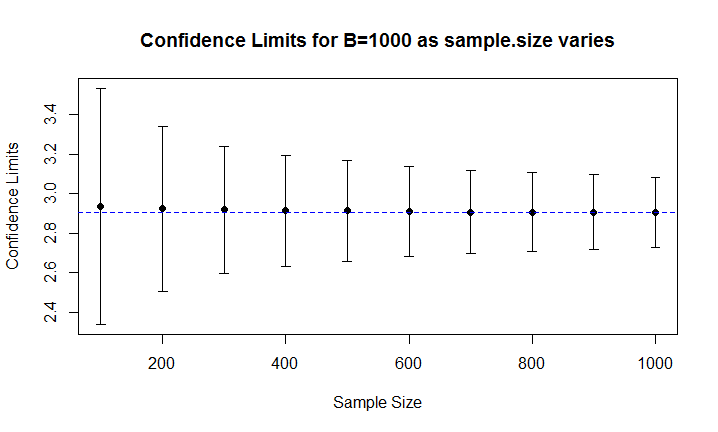
yplus=c(CI100sample[3,1],CI200sample[3,1],CI300sample[3,1],CI400sample[3,1],CI500sample[3,1],CI600sample[3,1],CI700sample[3,1],CI800sample[3,1],CI900sample[3,1],CI1000sample[3,1]),

yminus=c(CI100sample[1,1],CI200sample[1,1],CI300sample[1,1],CI400sample[1,1],CI500sample[1,1],CI600sample[1,1],CI700sample[1,1],CI800sample[1,1],CI900sample[1,1],CI1000sample[1,1]),

main = "Confidence Limits for B=1000 as sample.size varies",xlab = "Sample Size",ylab = "Confidence Limits")

title( main = "Confidence Limits for B=1000 as sample.size varies")

abline(h=lambda.hat,col="blue",lty=2)



# From the graph of confidence limits for sample size = 100 as B varies, we can conclude that

# the confidence intervals almost remain the same regardless of how B varies. There is only slightly

# change for different B. And for each B, the estimate of lambda.hat (mean) is very close to

# the true value of lambda.hat.

# From the graph of confidence limits for B = 1000 as sample size varies, we can conclude that

# the confidence intervals become narrower as sample size increases. And as sample size increases,

# the estimate of lambda.hat (mean) also gets closer to the true value of lamda.hat.

# 4a. (5) In 1961 John Tukey wrote an article called The Future of Data Analysis

# (it is uploaded in moodle). Some people say it is prophetic regarding the

# field of Data Science today. Do you agree or disagree? Why or why not? (Please

# keep your answer less than 500 words).

# John Tukey’s article titled The Future of Data Analysis is prophetic in some aspects regarding

# the field of Data Science. But there exist some limitations as well.

# For instance, his emphasis on judgement is critical to establish directions for any field.

# In this case, in Data Science field, he argued that evaluation of data analysis should be based on

# (1) experience of particular field of subject from which the data come from;

# (2) broad experience with how particular techniques can be utilized in various fields of application;

# (3) abstract results whether can be obtained by mathematical proofs.

# Those judgements are practical in today’s data analysis because different kinds of data analysis

# can be made towards heterogeneous data (qualitative and quantitative data) from multiple subjects.

# For history, maybe we are more interested in the trend, special events and causal relationships

# in the data; but for chemistry, we may more curious about structures of materials and reaction principles

# based on the data. The principal purposes for data analysis and the knowledge that we hope to discover

# are various among distinct subjects. Effective data analysis should be good at satisfying the needs of

# researchers in different subjects. This judgement is being considered in today’s data science

# so John Tukey made a good prediction. In addition, some common mathematical techniques are widely used

# in today’s data analysis such as regression model, hypothesis testing and correlation analysis.

# The trend prediction is a common analysis among many subjects and it also requires mathematical proofs.

# In general, the judgements that John proposed are prophetic in today’s data science.

# However, there are some limitations in John’s article. Nowadays as the development of data mining,

# machine learning, and natural language processing, even in digital humanities, text classification

# and word trend can be easily applied during the data analysis. But John did not mention the

# interdisciplinary cross and combination in computer science techniques towards multiple fields

# and internet of things.

# 4b. (5) Relate the article to the Life Cycle of Data discussion from class.

# You may wish to choose an example or idea from the article and clearly explore how it

# relates to the Life Cycle of Data. (Please keep your answer less than 500 words).

# John Tukey wrote in his paper that regression was one of the critical parts in factor

# analysis since regression techniques always offer hopes of learning more from less data

# than do variance-component techniques.

# One of the major challenges in data life cycle assessment is the availability and quality

# of data used to develop models and to make appropriate recommendations. The establishment

# of model based on original data can lead to further analysis and knowledge discovery procedure

# during the Life Cycle of Data.

# Approximations and assumptions are often made if appropriate data are not readily available.

# However, these proxies may introduce uncertainty into the results. A regression model

# framework may be employed to assess missing data in data life cycle assessment. (Steinmann et al., 2014)

# The regression model during the data analysis section performs like a bridge between raw data

# and reusable information. The coefficients, standard errors and outliers of the regression

# model can help to evaluate our initial hypothesis and draw empirical conclusions. The information

# about the regression effectiveness can be compared among different datasets, and we can discover

# new knowledge based on them. And this new knowledge can be transmitted into next section during

# the data life cycle such as data transformation and presentation. So regression analysis is

# a critical tache in the Life Cycle of Data.

# Steinmann, Z. J. N., Venkatesh, A., Hauck, M., Schipper, A. M., Karuppiah, R., Laurenzi, I. J., & Huijbregts, M. A. J. (2014).

# How to address data gaps in life cycle inventories: a case study on estimating CO2 emissions from coal-fired electricity plants on a global scale.

# Environmental Science & Technology, 48(9), 5282–5289. <https://doi.org/10.1021/es500757p>