Group2\_HW1\_Submision\_Brooke-Sean-Faizaan

2025-08-29

## Solution 1

options(repos = c(CRAN = "https://cran.rstudio.com"))  
# Make sure that this information is installed and applied beforehand  
install.packages("ISLR")

## Installing package into 'C:/Users/X/AppData/Local/R/win-library/4.5'  
## (as 'lib' is unspecified)

## package 'ISLR' successfully unpacked and MD5 sums checked  
##   
## The downloaded binary packages are in  
## C:\Users\X\AppData\Local\Temp\RtmpYHZuxB\downloaded\_packages

install.packages("patchwork")

## Installing package into 'C:/Users/X/AppData/Local/R/win-library/4.5'  
## (as 'lib' is unspecified)

## package 'patchwork' successfully unpacked and MD5 sums checked  
##   
## The downloaded binary packages are in  
## C:\Users\X\AppData\Local\Temp\RtmpYHZuxB\downloaded\_packages

library(patchwork)  
library(ISLR)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

dim(College) # rows, cols

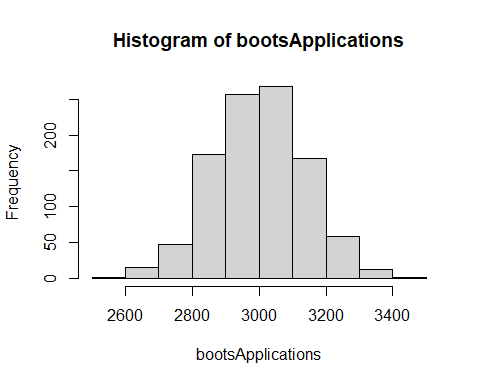
## [1] 777 18

help(College)

## starting httpd help server ...

## done

apps<-College$Apps  
#apps  
  
set.seed(1337)  
  
# setting up an empty var for use later.   
bootsApplications<-NULL  
  
# writing as dplyr so we could learn piping -- easier ways with just sample()  
library(dplyr)  
for(j in 1:1000)  
{  
 # in this case, the '.' is specifically piping "Apps" into those positions.   
 # so we're sampling (apps, len(apps), replacement = true)  
 # why we don't use mean(.) here is because mean already  
 # expects our sample from apps. So mean(.) and mean() are identical  
 my\_samp=apps%>%sample(.,length(.),replace=TRUE)%>%mean()  
 # Once we grab a single sample   
 bootsApplications<-c(bootsApplications,my\_samp)  
}  
hist(bootsApplications)



quantile(bootsApplications,c(.1,.9))

## 10% 90%   
## 2831.907 3179.010

## Solution 2

# WE are starting our our System time to get our init UTC time format.   
startTime <- Sys.time()  
boots1 <- NULL  
boots2 <- NULL  
boots3 <- NULL  
  
# Creating a vector "seeds" to hold our 3 random seed values. This only gets  
# run 1 time per execution and seeds is reused for our 5,000 samples later on.  
# Our code here uses the samples function to grab a random value between  
# 1 and 5,000 --- then does this 3 times exactly.   
seeds <- sample(1:5000,3)   
  
set.seed(seeds[1])  
for(j in 1:1000)  
{  
 my\_samp=apps%>%sample(.,length(.),replace=TRUE)%>%mean()  
 boots1<-c(boots1,my\_samp)  
}  
  
set.seed(seeds[2])  
for(j in 1:1000)  
{  
 my\_samp=apps%>%sample(.,length(.),replace=TRUE)%>%mean()  
 boots2<-c(boots2,my\_samp)  
}  
  
set.seed(seeds[3])  
for(j in 1:1000)  
{  
 my\_samp=apps%>%sample(.,length(.),replace=TRUE)%>%mean()  
 boots3<-c(boots3,my\_samp)  
}  
  
# Printing the combined data frame for boots 1,2,3 so we can verify our data.  
df <- data.frame(boots1,boots2,boots3)  
#print(df)  
  
boots4 <- NULL  
boots5 <- NULL  
boots6 <- NULL  
# Note: We had stored our seeds into the vector "seeds" so now we are grabbing  
# each seed 1,2,3 and placing those back into new samples @ 5,000 samples.   
set.seed(seeds[1])  
for(j in 1:5000)  
{  
 my\_samp=apps%>%sample(.,length(.),replace=TRUE)%>%mean()  
 boots4<-c(boots4,my\_samp)  
}  
  
set.seed(seeds[2])  
for(j in 1:5000)  
{  
 my\_samp=apps%>%sample(.,length(.),replace=TRUE)%>%mean()  
 boots5<-c(boots5,my\_samp)  
}  
  
set.seed(seeds[3])  
for(j in 1:5000)  
{  
 my\_samp=apps%>%sample(.,length(.),replace=TRUE)%>%mean()  
 boots6<-c(boots6,my\_samp)  
}  
  
# Grab our final UTC time after execution, then minus, then print the result.  
endTime <- Sys.time()  
elapsed <- endTime - startTime  
# NOTE: we are doing a version of Concatenation, but with strings on numbers, hense  
# paste  
print(paste("Main Calc Code: Time Elapsed - ",elapsed))

## [1] "Main Calc Code: Time Elapsed - 0.96312689781189"

## Graphing Solution 2

So now that we have the main calculation and have got all of our data, let’s graph it!

We’re going to do that with a mix of ggplot2 to create our hists with some colour and patchwork to put it all into 1 image for easy of use.

NOTE: that we are using bins of size 30 here for both and colouring each of them differently to show some differentiation between the graphs.

We are also only graphing boots1 vs boots 4, but the code can be extended to any of the combinations.

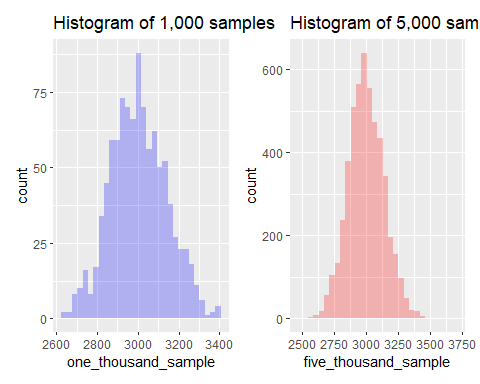
library(ggplot2)  
library(patchwork)  
startTime <- Sys.time()  
quantile(boots1,.1,.9)

## 10%   
## 2836.595

quantile(boots4,.1,.9)

## 10%   
## 2830.341

# ----------------------------------  
p1 <- ggplot(data.frame(one\_thousand\_sample = boots1), aes(x = one\_thousand\_sample)) +  
 geom\_histogram(bins = 30, fill = "blue", alpha = 0.25) +  
 labs(title = "Histogram of 1,000 samples")  
  
# Make graph 2~ 5,000 samples   
p2 <- ggplot(data.frame(five\_thousand\_sample = boots4), aes(x = five\_thousand\_sample)) +  
 geom\_histogram(bins = 30, fill = "red", alpha = 0.25) +  
 labs(title = "Histogram of 5,000 samples")  
# ----------------------------------  
# Combine both hists with patchwork  
p1 + p2



endTime <- Sys.time()  
elapsed <- endTime - startTime  
print(paste("Graphing Code: Time Elapsed - ", elapsed))

## [1] "Graphing Code: Time Elapsed - 0.180788040161133"

## Hists We Produced - Our Comments

Our observations from the first histogram is that it is rather noisy and jagged. The 1,000 sample looks mildly left tailed and has outliers compared the the 5,000 sample. 1,000 samples seems to not be enough because the distribution is a bit left leaning, whereas the 5,000 samples gives us a nicer bell shape. The left almost seems as if it has missing parts. The right histogram (5,000) looks much smoother than the left. It appears more uniform and symmetrical and gives us more confidence. We do wonder if there is a correlation between more samples and uniformity or if the results are similar but look different.

We are astonished that the computational time is so quick. We’re not sure if this is CPU bound or not. IE Throwing more hardware would decrease comp time.

## SOLUTION 3

# Let's start by grabbing just the private data and looking at it.   
isPrivateOrPublic <- College$Private  
#isPrivateOrPublic  
  
# Set base values to 0 so there is nothing from previous runs of this code  
bootPriOrPub <- NULL  
privateCount <- 0  
pubCount <- 0  
  
# we know our dataset is 777 in size, In this case, we were not able to sample  
# when the data was Alphas, so we completed a quick transpose to 0 for 1  
sampleSet <- NULL  
for(j in 1:777)  
{  
 if(isPrivateOrPublic[j] == "No")  
 {  
 sampleSet <- c(sampleSet,0)  
 }  
 if(isPrivateOrPublic[j] == "Yes")  
 {  
 sampleSet <- c(sampleSet,1)  
 }  
}  
  
# Let's now sample from the new "Sample Set" and do it 5,000 times  
bootsPriOrPub <- NULL  
for(j in 1:5000)  
{  
 my\_samp=sampleSet%>%sample(.,length(.),replace=TRUE)%>%mean()  
 bootsPriOrPub<-c(bootsPriOrPub,my\_samp)  
}  
  
summary(sampleSet)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0000 0.0000 1.0000 0.7272 1.0000 1.0000

# From research, the t.test method will create a large list of data, one of which  
# is our double sided p.value test.   
pTest <- t.test(sampleSet)  
# We are then calling into the 3rd position of that list to grab the P-value  
pTest[3]

## $p.value  
## [1] 4.579363e-221

At a 0.05 significance level, our p value is less and therefore we reject the hypothesis. The visibility into this data set is not very high due to limited data. 777 data points in the set is very minimal compared to the entirety of the national scale. It is surely not enough to make conclusions, as more data is likely missing than what we interpreted. It’s difficult to assume that 777 data points could scale to a hypothesis about the entire country rather than just a region. If this were just a county, maybe we could consider.