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\RtmpQ79UZJ\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\RtmpQ79UZJ\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

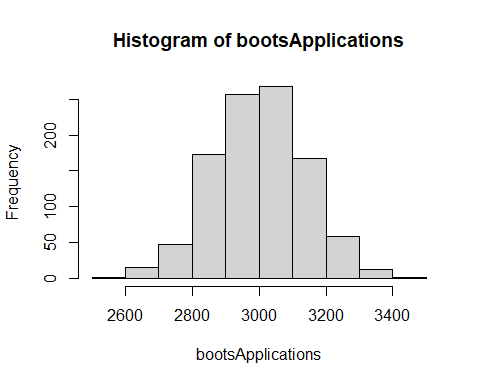
## [1] 777 18

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

At an 80% confidence level, we see great clustering around the 3,000 mark for our applications to colleges in our dataset. With a higher rise at the 2,800 mark and 3,200 mark, we might be able to say that we have a bit of a concentrated number for most colleges between this range. The rest are not outlines, but with more data, I might expect our mid point to rise higher and fall off steeper.

## Solution 2

startTime <- Sys.time()  
# 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)  
endTime <- Sys.time()  
elapsed <- endTime - startTime  
print(paste("1,000 sample Calc Code: Time Elapsed - ",elapsed))

## [1] "1,000 sample Calc Code: Time Elapsed - 0.168615102767944"

Once our 1,000 samples has been run 3 times, we have our seeds stored for use in our 5,000 sample distributions that we will see below.

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("5,000 sample Calc Code: Time Elapsed - ",elapsed))

## [1] "5,000 sample Calc Code: Time Elapsed - 1.09627199172974"

## 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.

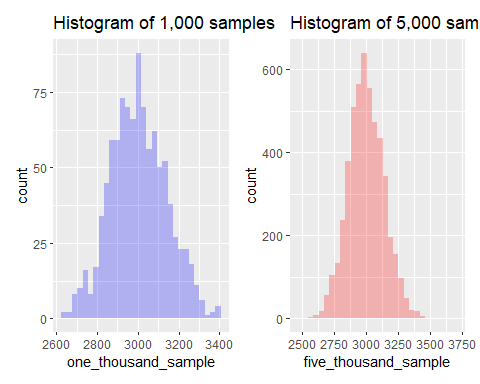
library(ggplot2)  
library(patchwork)  
startTime <- Sys.time()  
quantile(boots1,.1,.9)

## 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 - 0.273139953613281"

## 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  
  
# tapply ( x, index, mean/median/other)  
# do calcs on X, for each instance of Index, then compute the mean on X.   
appsGroupedByPrivate<-tapply(College$Apps,College$Private,mean)  
appsGroupedByPrivate

## No Yes   
## 5729.920 1977.929

my\_stat<-appsGroupedByPrivate[1] - appsGroupedByPrivate[2]  
data.frame(my\_stat)

## my\_stat  
## No 3751.991

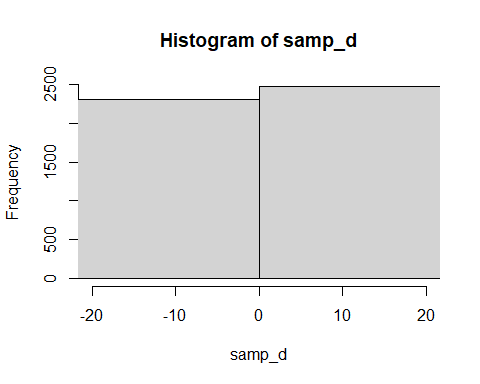
TApply allows us to group two cols together and then do an execution on 1 of those. In our usecase, we are spinning our apps data around our ‘Yes’ and ‘No’ per entry of yes and no. This is why y/n is our headers and apps is our data in the col. Once we have these, tapply is set to use the mean to get the “average” of all grouping of ‘yes’ and ‘no’. This only works one way as we can’t get the mean of alpha values.

minus our stats gives us what is the “remainder” which tells us which is more than the other in terms of college applications from private vs public schools. We observe still mostly 'No' or Private applications overwhelming the public averages per College.

##Info-Industrial>0 (Alt Hyp)  
## Info-Industiral=0 (Null Hyp)  
## We want to generatee the null hypothesis that the two means are equal  
## We want to know do info make more than industrial (avg higher)  
set.seed(660)  
samp\_d<-NULL  
dim(College) # 777 -- so we use this in our sampling

## [1] 777 18

for(j in 1:10000){  
 NewDat<-data.frame(appers=College$Apps,priv=sample(College$Private, 777))  
 my\_stat\_boot<-tapply(NewDat$appers,NewDat$priv,mean)  
 samp\_d<-c(samp\_d,my\_stat\_boot[2]-my\_stat\_boot[1])  
 ## ^^ null distro ^^  
}  
#df<- data.frame(samp\_d)  
#df  
  
hist(samp\_d,xlim=c(-20,20))  
# WE CANNOT CALL THIS BY ITSELF   
abline(v=my\_stat)



my\_stat

## No   
## 3751.991

# div by the number of samples taken  
sum(samp\_d>=my\_stat)/10000

## [1] 0

# there are 0 times our last sets didn't show up in the distro

Out end value for P value is 0. Because that is the case, we will reject the null as it’s the most correlated hypothesis scenario possible. Correlation however does not equal causation and we’re not measuring external variables, but based on what we were given, we can say that our test is less than 0.05.