# BAS 471 Spring 2023 Homework on Unit 9 Parts 5-6 - Comparing Averages and Bootstrapping

Ryan Curling

5/4/2023

---

## Note: these are your homework problems

Reminder about collaboration policy. You can develop a common set of R code with you and your friends. However, anything that is written interpretation, i.e., anything that follows a **Response:** needs to be written up in your own words. Homeworks that look to be near copy/pastes of each other will receive substantially reduced credit.

---

### Question 1 - Drivers of Movie Success

The COVID-19 pandemic has significantly impacted the movie industry, potentially altering its landscape permanently, despite its ongoing recovery. This comes after years of struggling due to the increasing emphasis on profit and changing audience expectations. To tackle this challenge, studios have been utilizing artificial intelligence, as learned in BAS 320/474, to analyze the elements that attract audiences and generate profits.

As Hollywood studios tend to avoid taking big risks, any insights into the factors that contribute to a movie's success are highly valuable. In this context, we aim to investigate the main drivers behind the success of movies using the `MOVIE` dataframe in `BAS471Sp23Unit9Parts4-5.RData`.

This dataset includes information on 1271 movies released between 1993 and 2019, with budgets ranging from 20 to 50 million dollars and with MPAA ratings of PG, PG-13, or R. The data includes various characteristics such as rating, runtime, genre, and cinematic universe, as well as information on distributors, stars, and directors. Moreover, performance indicators such as return on investment (ROI), critic and audience ratings, and tomatometer scores are also available.

You know the drill:

- Make sure the .RData file is in your 471 folder
- Make sure to do a Save As on this .Rmd file and forcibly put it in your 471 folder
- Make sure your working directory is your 471 folder too

We can leverage this data to give general suggestions on the types of movies to create in order to optimize their potential for success. Let's focus on `ROI` and `AudienceRating`.

- ROI is the return on investment. ROI measures the amount of profit generated in relation to the amount of money invested in the production of the movie (i.e., its budget). A value of 0 means it only made back its budget (but no additional profit). A value of 1.5 means that the movie not only made back its budget, but that the profit was 1.5x its budget. A value of -0.5 means it only made back 1/2 its budget. A value of -1 means all money was lost.

- Audience Rating is a score from 0-1 with larger values being more favorable.

a) Your first inquiry is determining whether there is a statistically significant difference in the average `ROI` or `AudienceRating` across `Universe`. Movies in the "Yes" group take place within a cinematic universe like Star Wars, Harry Potter, or Marvel, etc. Movies in the "No" group do not. When comparing two groups, the first question you must answer is "Are these paired samples or independent samples?". How should we treat these samples: paired or independent? Explain.

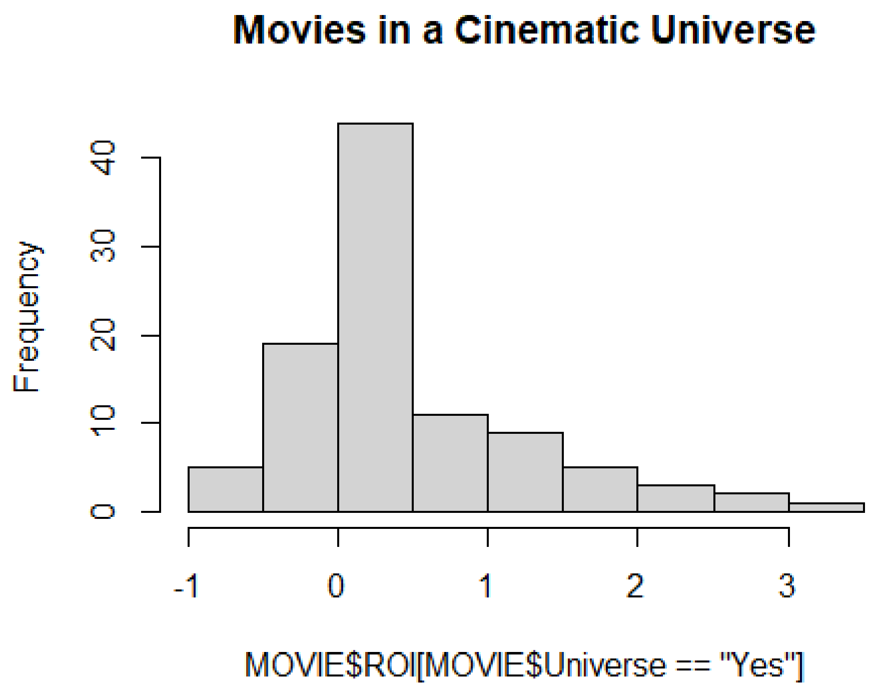
**Response: We should treat this sample as independent samples. This is because both 'Yes' and 'No' responses in 'Universe' are not related or paired in anyway to the observation sin the opposite group.**

```
load("BAS471Sp23Unit9Parts4-5.RData")
table(MOVIE$Universe)
##
##   No  Yes
## 1172   99
```

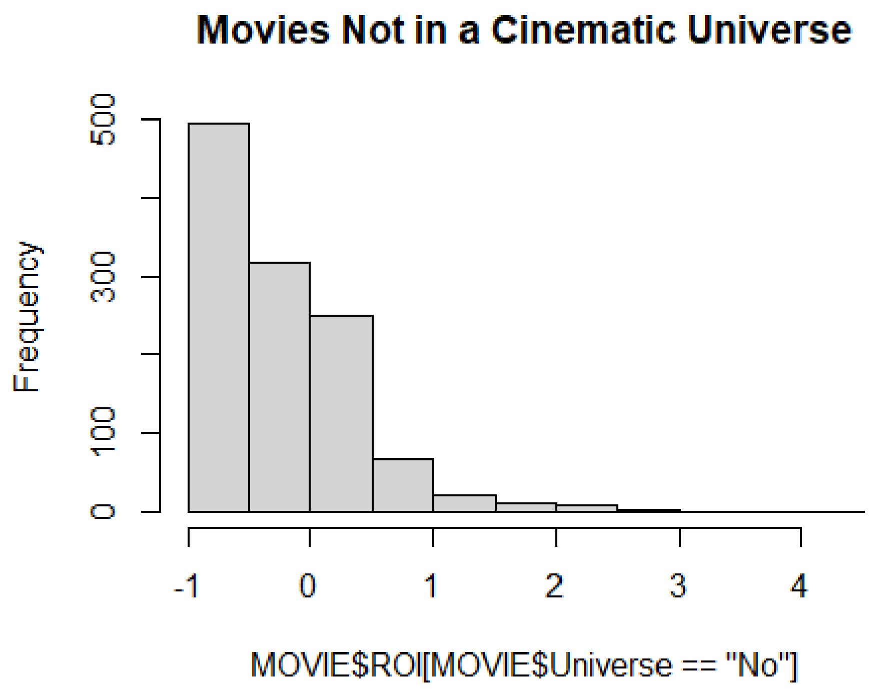
b) The second question you must answer when comparing two groups is whether the Central Limit Theorem is in effect. If it is, then `t.test` can quickly produce a 95% confidence interval and you can determine whether the difference in averages is statistically significant. Make histograms of the `ROI` for movies in a cinematic universe and for movies not in a cinematic universe. Looking at the histograms and referring to sample sizes from Q1a, is the Central Limit Theorem in effect if we want to compare average `ROI`s? Explain. Spoiler alert, the CLT is in effect when comparing `AudienceRating`.

**Response: The CLT is not in effect for the ROI in the cinematic universe, this is because the data is highly skewed. Also with only 99 movies in the 'Yes' response, our sample size is not high enough to say the CLT theorem applies.**

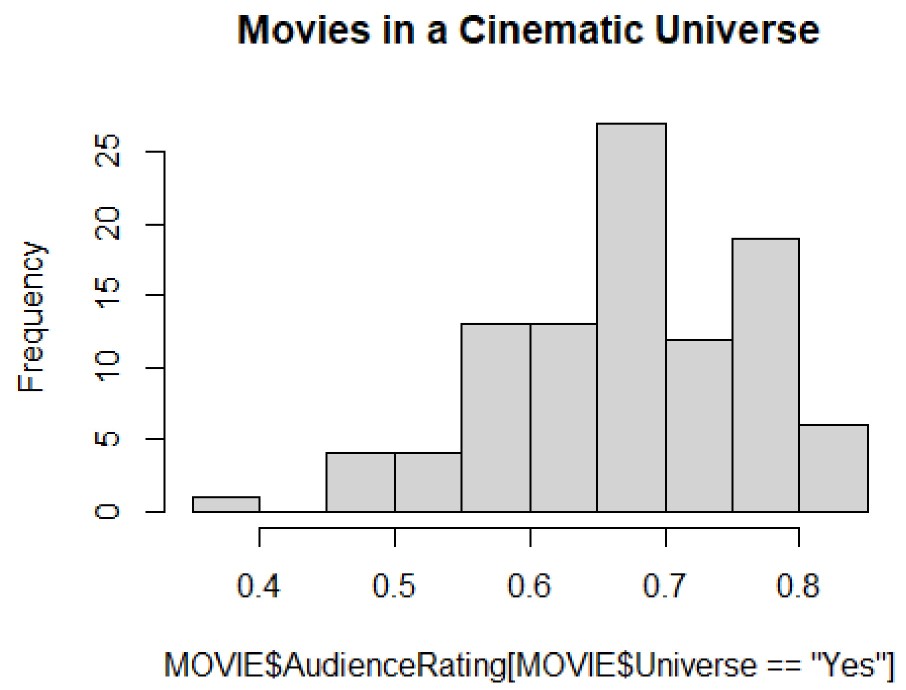
```
hist(MOVIE$ROI[MOVIE$Universe=="Yes"], main="Movies in a Cinematic Universe")
```



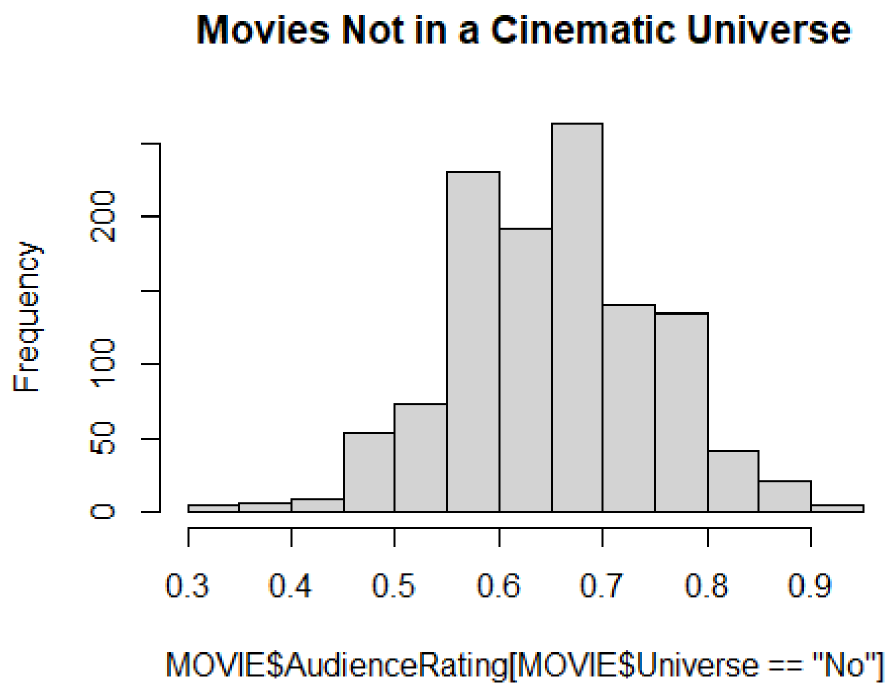
```
hist(MOVIE$ROI[MOVIE$Universe=="No"], main="Movies Not in a Cinematic
Universe")
```



```
hist(MOVIE$AudienceRating[MOVIE$Universe=="Yes"], main="Movies in a Cinematic
Universe")
```



```
hist(MOVIE$AudienceRating[MOVIE$Universe=="No"], main="Movies Not in a
Cinematic Universe")
```



c) Report the average ROI for movies in a cinematic universe and for movies not in a cinematic universe. Is the difference in averages statistically significant? Explain. If the difference is statistically significant, explain whether the difference is large enough to be meaningful and impactful.

**Response: Yes, this difference in averages is statistically significant. We know this because zero is not contained in the confidence interval. Cohen's d is greater than 0.8, therefore we can conclude the difference is large enough to be meaningful and impactful.**

```
# Subset the data by cinematic universe
universe_yes <- MOVIE[MOVIE$Universe == "Yes", ]
universe_no <- MOVIE[MOVIE$Universe == "No", ]

# Calculate the mean ROI for each group
mean_roi_yes <- mean(universe_yes$ROI)
mean_roi_no <- mean(universe_no$ROI)

mean_roi_yes
## [1] 0.4666667
mean_roi_no
## [1] -0.2450597

t_test <- t.test(universe_yes$ROI, universe_no$ROI)
```

```
t_test$p.value
## [1] 8.617711e-14
t_test$conf.int
## [1] 0.5469441 0.8765087
## attr(,"conf.level")
## [1] 0.95

#Is difference meaningful? Calc effect size

d <- abs(mean_roi_yes - mean_roi_no) / sd(MOVIE$ROI)
d
## [1] 1.08563
```

d) Repeat part (c) but for average Audience Rating.

**Response: Since zero,is not contained in the confidence interval we can conclude the difference in averages between the two groups to be statistically significant. Cohen's d is equal to .205 in this case, so the difference can be considered meaningful (but barely). In practice, I would be hesitant to say the difference was that meaninnful.**

```

yes_univ_ar <- subset(MOVIE, Universe=="Yes")$AudienceRating
no_univ_ar <- subset(MOVIE, Universe=="No")$AudienceRating

# Mean of AudienceRating
mean_univ_ar <- mean(yes_univ_ar)
mean_no_univ_ar <- mean(no_univ_ar)

# 95% confidence interval using t-test assuming independent samples
ttest_ar <- t.test(yes_univ_ar, no_univ_ar)
ci_ar <- ttest_ar$conf.int

ttest_ar
##
##  Welch Two Sample t-test
##
## data:  yes_univ_ar and no_univ_ar
## t = 2.0402, df = 116.91, p-value = 0.04358
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.0005756936 0.0387441257
## sample estimates:
## mean of x mean of y
## 0.6761616 0.6565017
ci_ar
## [1] 0.0005756936 0.0387441257
## attr(,"conf.level")
## [1] 0.95

#Cohen's d
```

```
d <- abs(mean_univ_ar - mean_no_univ_ar) / sd(MOVIE$AudienceRating)
d
## [1] 0.20517
```

e) Most movie characteristics have more than two levels, e.g. Rating, Runtime, MainGenre, Director, etc. Provide the output from `aggregate` that gives the average `ROI` for the three levels of `Rating`. Without making a connecting letters report, explain how you know that there are *no* statistically significant differences in average ROI across these 3 MPAA-ratings.

**Note:** This step was unfortunately omitted from the lecture/notes. Look at the `Pr(>F)` in the analysis of variance. If this is *less* than 0.05, there *are some* statistically significant differences between groups (however, the analysis of variance doesn't tell us *which* groups are different). ONLY WHEN the `Pr(>F)` is less than 0.05 do we continue the analysis and look at the connecting letters. It is possible that `Pr(>F)` is less than 0.05 (indicating some differences) but all connecting letters will be the same (meaning the data can't discern differences after all).

**Response: Not statistically significant.**

```
aggregate(ROI ~ Rating, data = MOVIE, FUN = mean)
##   Rating        ROI
## 1     PG -0.1857459
## 2   PG13 -0.1968588
## 3      R -0.1846167

anova(lm(ROI ~ Rating, data = MOVIE))
## Analysis of Variance Table
##
## Response: ROI
##             Df Sum Sq Mean Sq F value Pr(>F)
## Rating       2   0.04 0.02188  0.0508 0.9504
## Residuals 1268 545.80 0.43044
```

f) Provide the output from `aggregate` that gives the average `AudienceRating` for each of the 8 levels of `RunTime`. The ANOVA reveals that there are *some* statistically significant differences, so produce a connecting letters report to find them. Note: the levels give ranges of runtimes. A number next to a parenthesis is NOT included in the range, but a number next to a bracket is included. For example, (100,105] mean runtimes *greater* than 100 and *at most* 105 (does not include 100 but does include 105).

```
library(multcompView)

aggregate(AudienceRating ~ RunTime, data = MOVIE, FUN = mean)
##     RunTime AudienceRating
## 1    (0,90]      0.5968750
## 2   (90,95]      0.6058156
## 3  (95,100]      0.6266667
## 4 (100,105]      0.6419883
## 5 (105,110]      0.6572619
## 6 (110,120]      0.6890583
## 7 (120,130]      0.7327152
## 8 (130,300]      0.7457831

library(multcomp)
## Loading required package: mvtnorm
## Loading required package: survival
##
## Attaching package: 'survival'
## The following object is masked from 'package:boot':
##
##     aml
## Loading required package: TH.data
## Loading required package: MASS
##
## Attaching package: 'TH.data'
## The following object is masked from 'package:MASS':
##
##     geyser

AOV <- aov(AudienceRating ~ RunTime, data = MOVIE)
summary(AOV)
##                Df Sum Sq Mean Sq F value Pr(>F)
## RunTime         7  2.894  0.4135   59.56 <2e-16 ***
## Residuals    1263  8.767  0.0069
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


TUKEY <- TukeyHSD(AOV)
TUKEY
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = AudienceRating ~ RunTime, data = MOVIE)
##
## $RunTime
##                            diff          lwr        upr     p adj
## (90,95]-(0,90]      0.008940603 -0.020276411 0.03815762 0.9832560
## (95,100]-(0,90]     0.029791667  0.002086501 0.05749683 0.0248471
## (100,105]-(0,90]    0.045113304  0.017291962 0.07293465 0.0000265
## (105,110]-(0,90]    0.060386905  0.032445745 0.08832806 0.0000000
## (110,120]-(0,90]    0.092183296  0.065976792 0.11838980 0.0000000
## (120,130]-(0,90]    0.135840232  0.107142084 0.16453838 0.0000000
## (130,300]-(0,90]    0.148908133  0.114692369 0.18312390 0.0000000
## (95,100]-(90,95]    0.020851064 -0.007810061 0.04951219 0.3470576
## (100,105]-(90,95]   0.036172701  0.007399259 0.06494614 0.0035406
## (105,110]-(90,95]   0.051446302  0.022556992 0.08033561 0.0000021
## (110,120]-(90,95]   0.083242693  0.056027537 0.11045785 0.0000000
## (120,130]-(90,95]   0.126899629  0.097277551 0.15652171 0.0000000
## (130,300]-(90,95]   0.139967530  0.104973211 0.17496185 0.0000000
## (100,105]-(95,100]  0.015321637 -0.011915346 0.04255862 0.6822044
## (105,110]-(95,100]  0.030595238  0.003235879 0.05795460 0.0161959
## (110,120]-(95,100]  0.062391629  0.036806340 0.08797692 0.0000000
## (120,130]-(95,100]  0.106048565  0.077916558 0.13418057 0.0000000
## (130,300]-(95,100]  0.119116466  0.085374139 0.15285879 0.0000000
## (105,110]-(100,105] 0.015273601 -0.012203398 0.04275060 0.6954231
## (110,120]-(100,105] 0.047069992  0.021358945 0.07278104 0.0000009
## (120,130]-(100,105] 0.090726928  0.062480499 0.11897336 0.0000000
## (130,300]-(100,105] 0.103794828  0.069957046 0.13763261 0.0000000
## (110,120]-(105,110] 0.031796391  0.005955741 0.05763704 0.0048167
## (120,130]-(105,110] 0.075453327  0.047088878 0.10381778 0.0000000
## (130,300]-(105,110] 0.088521228  0.054584864 0.12245759 0.0000000
## (120,130]-(110,120] 0.043656936  0.016999583 0.07031429 0.0000207
## (130,300]-(110,120] 0.056724837  0.024201779 0.08924789 0.0000039
## (130,300]-(120,130] 0.013067901 -0.021494392 0.04763019 0.9459580

multcompLetters4(AOV,TUKEY)
## $RunTime
## (130,300] (120,130] (110,120] (105,110] (100,105]  (95,100]   (90,95] 
(0,90] 
##       "a"       "a"       "b"       "c"      "cd"      "de"      "ef" 
"f"
```

For the following, answer "yes" or "no", then briefly explain:

- Can the data discern a difference in the average audience rating of movies with runtimes of (100,105] (average 0.642) and of movies with runtimes of (90,95] (average 0.606)?

**Response: Yes, the data can discern a difference between the two groups. We know this because the two groups do not share any letters in common. Therefore there is a statistically significant difference in groups.**

- Does the data indicate a statistically significant difference in the mean audience ratings between movies with runtimes falling in the range of (95,100] (with an average of 0.627) and those with runtimes in the range of (100,105] (with an average of 0.642)?

**Response: No, since they share a letter in common there is no statistically significant differences between the groups.**

- Movies with runtimes greater than 130 have the highest average audience rating (0.746). Are other runtimes "statistically tied" for this highest average? Why or why not? If so, which?

**Response: Yes the runtime group of [120,130] is statistically tied to the 130 and over group. This is because contain ‘a’, and therefore there is not a statiscally signifcant difference between them.**

g) (Bonus) In analytics, we are often interested in the “biggest drivers” of some process of interest (donating, churning, buying, etc.). Here, we want to know the “biggest drivers” of `ROI` and of `AudienceRating`. A “big” driver will have large and statistically significant differences in averages across the driver’s possible values.

One way to “score” a driver is to fit a multiple linear regression model (BAS 320) using the driver as the sole predictor, e.g. `summary( lm(ROI ~ Rating,data=MOVIE) )`. If you recall, a multiple linear regression model also allows you to determine if there are any statistically significant differences in the average value of “y” (the process you’re studying, e.g. ROI) across levels of a categorical predictor (the driver). The “Adjusted R-Squared” of the model allowed us to quantify, at least roughly, what percentage of the variation in “y” can be attributed to the driver. Higher adjusted R-Squared = higher score = stronger driver = more variation in average “y” across the driver’s levels.

Find the “scores” (Adjusted R-Squared) of Rating, Distribution, RunTime, TopBilledStar, MainGenre, SecondaryGenre, Director, Universe, and Month (9 total drivers), then provide a ranked list of the top 4 drivers (along with their scores) of ROI and of AverageRating (these will be separate lists). Note: appending `[9]$adj.r.squared` to the end of the `summary()` command above will directly extract the adjusted R-squared.

Obviously, studios would like to make a lot of money while also making audiences happy. What two drivers do you suggest studio executives focus on to make this happen? Explain.

**Response: Drivers to use: ROI = ‘Universe’ , AudienceRating = ‘RunTime’.**

```
# Fit models and extract adjusted R-squared
ROI_Rating_model <- lm(ROI ~ Rating, data = MOVIE)
summary(ROI_Rating_model)$adj.r.squared
## [1] -0.001496975

ROI_Distribution_model <- lm(ROI ~ Distribution, data = MOVIE)
summary(ROI_Distribution_model)$adj.r.squared
## [1] 0.002615502

ROI_RunTime_model <- lm(ROI ~ RunTime, data = MOVIE)
summary(ROI_RunTime_model)$adj.r.squared
## [1] 0.01447319

ROI_TopBilledStar_model <- lm(ROI ~ TopBilledStar, data = MOVIE)
summary(ROI_TopBilledStar_model)$adj.r.squared
## [1] 0.007593305

ROI_MainGenre_model <- lm(ROI ~ MainGenre, data = MOVIE)
summary(ROI_MainGenre_model)$adj.r.squared
```

```
## [1] 0.03690163

ROI_SecondaryGenre_model <- lm(ROI ~ SecondaryGenre, data = MOVIE)
summary(ROI_SecondaryGenre_model)$adj.r.squared
## [1] 0.0006863955

ROI_Director_model <- lm(ROI ~ Director, data = MOVIE)
summary(ROI_Director_model)$adj.r.squared
## [1] 0.007691949

ROI_Universe_model <- lm(ROI ~ Universe, data = MOVIE)
summary(ROI_Universe_model)$adj.r.squared
## [1] 0.08399707

ROI_Month_model <- lm(ROI ~ Month, data = MOVIE)
summary(ROI_Month_model)$adj.r.squared
## [1] 0.01458874

AudienceRating_Rating_model <- lm(AudienceRating ~ Rating, data = MOVIE)
summary(AudienceRating_Rating_model)$adj.r.squared
## [1] 0.006706253

AudienceRating_Distribution_model <- lm(AudienceRating ~ Distribution, data =
MOVIE)
summary(AudienceRating_Distribution_model)$adj.r.squared
## [1] 0.009127721

AudienceRating_RunTime_model <- lm(AudienceRating ~ RunTime, data = MOVIE)
summary(AudienceRating_RunTime_model)$adj.r.squared
## [1] 0.2440237

AudienceRating_TopBilledStar_model <- lm(AudienceRating ~ TopBilledStar, data
= MOVIE)
summary(AudienceRating_TopBilledStar_model)$adj.r.squared
## [1] 0.002122158

AudienceRating_MainGenre_model <- lm(AudienceRating ~ MainGenre, data =
MOVIE)
summary(AudienceRating_MainGenre_model)$adj.r.squared
## [1] 0.09996497

AudienceRating_SecondaryGenre_model <- lm(AudienceRating ~ SecondaryGenre,
data = MOVIE)
summary(AudienceRating_SecondaryGenre_model)$adj.r.squared
## [1] 0.05991825

AudienceRating_Director_model <- lm(AudienceRating ~ Director, data = MOVIE)
summary(AudienceRating_Director_model)$adj.r.squared
## [1] 0.001531445
```

```
AudienceRating_Universe_model <- lm(AudienceRating ~ Universe, data = MOVIE)
summary(AudienceRating_Universe_model)$adj.r.squared
## [1] 0.002240171

AudienceRating_Month_model <- lm(AudienceRating ~ Month, data = MOVIE)
summary(AudienceRating_Month_model)$adj.r.squared
## [1] 0.01752925
```

---

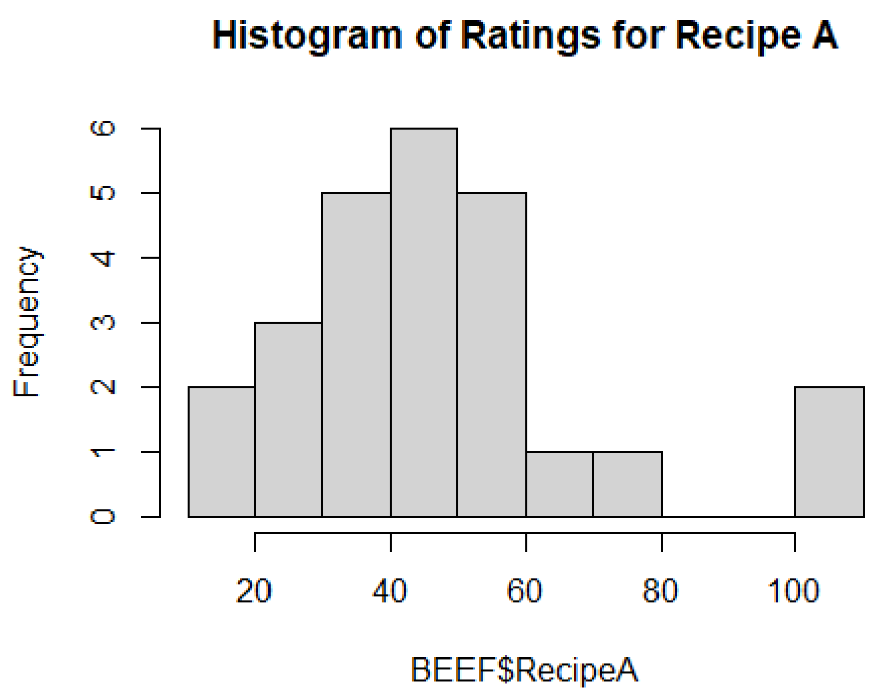
## Question 2

A chef wants to determine which of two recipes for Beef Bourguignon is better. The chef recruits 25 foodies and has them rate (on a scale of 0-100) the dishes on two sequential Tuesday nights. The `BEEF` dataset in the global environment (after loading in the .RData file) records each foodies' ratings (0-100) of recipes A and B.

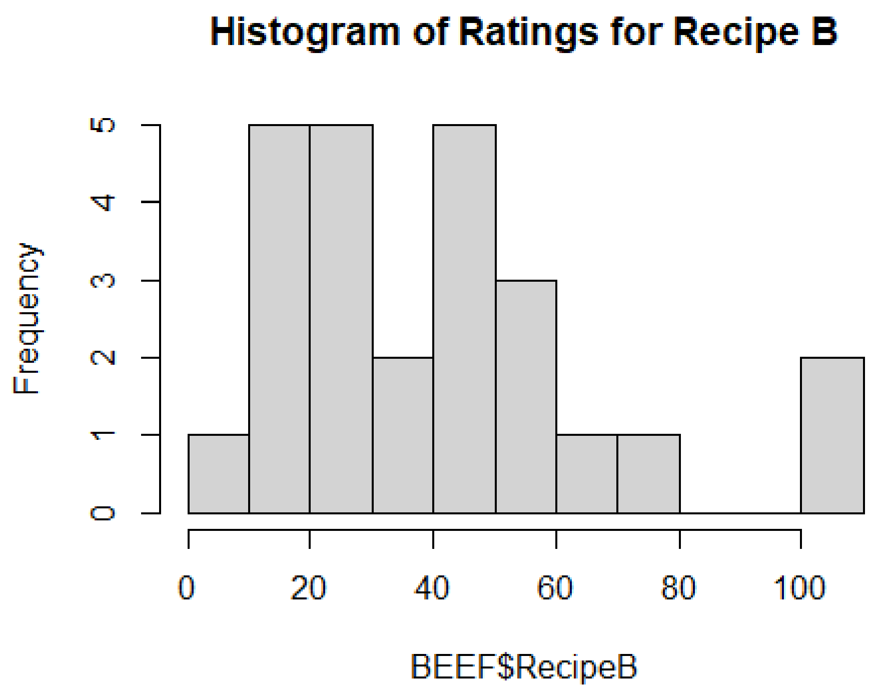
a) Can we invoke the Central Limit Theorem here to help create reliable confidence intervals for the averages of whatever probability distributions happen to be generating these samples? In other words, are the "large enough" sample size conditions met? Why or why not? Note: you'll need to make two histograms.

**Response: Although the sample size, is 25 we can not use the CLT. The minimum cutoff for slightly skewed is a smaple size of 25. However, I would say that this data is more than slightly skewed, which would require a sample size of 100 or more. Therefore, due to the insuffiencent sample size, we can not use the CLT.**

```
hist(BEEF$RecipeA, breaks = 10, main = "Histogram of Ratings for Recipe A")
```



```
hist(BEEF$RecipeB, breaks = 10, main = "Histogram of Ratings for Recipe B")
```



b) Are the ratings for Recipe A and B paired or independent samples? Explain.

**Response: Because the ratings of recipe A do not effect the ratings of B (and vice versa), we can classify these rarings as independent sample.**

c) Produce a 95% confidence interval for the "true" difference in average ratings of the recipes. Comment on the plausibility/reasonability that the difference in average ratings is 0.

**Response: The t-test below indicates that we can not reject the null hypothesis that the true difference in avg ratings between the recipes is 0. Since our confidence interval contains zero, this means the true difference in average rating of zero is plausible.**

```
t.test(BEEF$RecipeA,BEEF$RecipeB,paired = FALSE, conf.level = 0.95)
##
##  Welch Two Sample t-test
##
## data:  BEEF$RecipeA and BEEF$RecipeB
## t = 1.0586, df = 47.387, p-value = 0.2952
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -6.588278 21.228278
## sample estimates:
## mean of x mean of y
##     47.56     40.24
```

d) Run `t.test` the "wrong" way (i.e., if they are paired samples, do the independent sample test; if they are independent samples, do the paired test). Notice how the conclusion from (c) changes if you perform the wrong test!

```
t.test(BEEF$RecipeA,BEEF$RecipeB,paired = TRUE, conf.level = 0.95)
##
##  Paired t-test
##
## data:  BEEF$RecipeA and BEEF$RecipeB
## t = 6.1091, df = 24, p-value = 2.607e-06
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  4.846994 9.793006
## sample estimates:
## mean difference
##            7.32
```

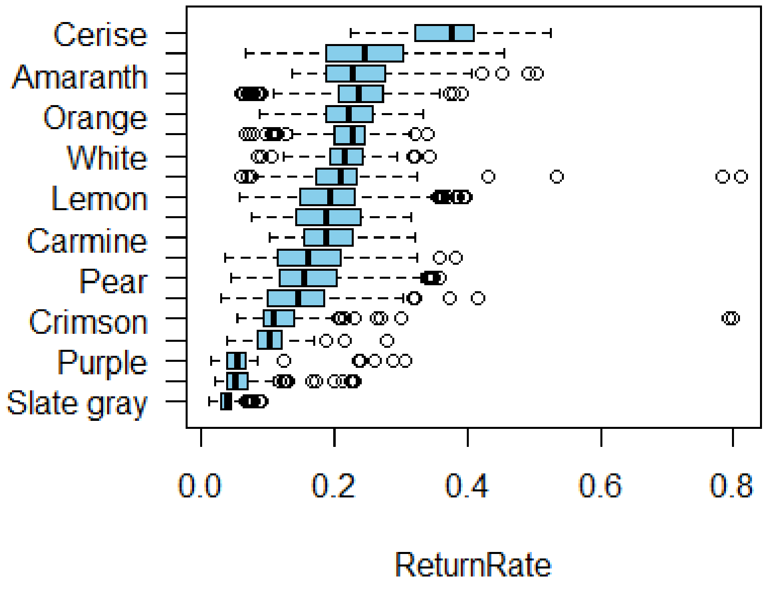
---

## Question 3 (bonus)

At Cannon Company, some items have high return rates while other items have low return rates. The data in `RETURNS` (in the saved .RData for this assignment) shows the weekly return rates (fraction of items in a product category that got returned that week) of different product categories (anonymized and referred to as Codenames).

Boxplots of the weekly return rates reveal that they vary quite a bit from week to week within a given Codename, and that Codenames can have vastly different average weekly return rates.

```r
#Run whole chunk as is
load("BAS471Sp23Unit9Parts4-5.RData")
#Boxplot
par(las = 1,mar=c(5,8,4,2)+0.1) ; boxplot( ReturnRate ~ Codename,
data=RETURNS, horizontal=TRUE,ylab="",col="skyblue"); par(las =
0,mar=c(5,5,4,2)+0.1)
```



a) Let’s compare the average return rate for Codenames “Lemon” and “White” (19.4% and 21.4%, respectively). Whenever we compare two groups, the first question we have to ask ourselves is: “are these paired samples or independent samples?” Assuming customers never purchase products from *both* Codenames in a given week, which type of samples are we dealing with? Explain.

**Response: Since they ‘never’ purchases from both Codenames in a week, these samples are independent from each other.**

b) Produce a 95% confidence interval for the difference in average return rates between “Lemon” and “White”. Can the data discern a difference in the “true” average return rates between these codenames? Why or why not? If the difference is statistically significant, interpret the values in the confidence interval. Note: a subset of the data with just these two codenames is provided for you.

**Response: Although the p-value is far below the 0.05 threshold, we can not say that there is a true difference in average return rates between btoh groups. This is because the confidence interval contains zero.**

```
TWO <- droplevels( subset( RETURNS, Codename %in% c("White","Lemon") ) )

t.test(ReturnRate~Codename , data=TWO, paired=FALSE, conf.level=0.95)
##
##  Welch Two Sample t-test
##
## data:  ReturnRate by Codename
## t = -5.3508, df = 250.57, p-value = 1.976e-07
## alternative hypothesis: true difference in means between group Lemon and
group White is not equal to 0
## 95 percent confidence interval:
##  -0.02733767 -0.01262751
## sample estimates:
## mean in group Lemon mean in group White
##           0.1943741           0.2143567
```

c) Codename “Sangria” has the 2nd highest average return rate (24.6%). Codename “Amaranth” has the 3rd highest (24.1%). Can the data discern a difference in the “true” average return rates between these groups? Explain. If the difference is statistically significant, interpret the values in the confidence interval.

**Response: No again, since the confidence interval contains zero we can not say the data can discern a difference in the “true” average return rates between these groups.**

```

# subset data to only include Sangria and Amaranth
TWO2 <- droplevels(subset(RETURNS, Codename %in% c("Sangria", "Amaranth")))
# perform two-sample t-test assuming independent samples
t.test(ReturnRate ~ Codename, data = TWO2, var.equal = TRUE)
##
##  Two Sample t-test
##
## data:  ReturnRate by Codename
## t = -0.6619, df = 1342, p-value = 0.5081
## alternative hypothesis: true difference in means between group Amaranth
and group Sangria is not equal to 0
## 95 percent confidence interval:
##  -0.017141651  0.008492499
## sample estimates:
## mean in group Amaranth  mean in group Sangria
##              0.2413373              0.2456619
```

d) Let's "rank" the return rates of six Codenames with "fairly typical" average return rates: "White", "Puce", "Lemon", "Cyan", "Carmine", "Amethyst". Produce a connecting letters report.

- Which of these six codenames has the highest average return rate from a statistical point of view (or is there a tie; list all groups tied for highest if so)?
- Which of these six codenames has the lowest average return rate from a statistical point of view (or is there a tie; list all groups tied for lowest if so)?
- Can the data discern a difference in the average return rates of "Puce" and "Carmine"? Explain.

**Responses: Since both groups share a letter in common, the data can not discern a difference in the average return rates of "Puce" and "Carmine".**

```
SIX <- droplevels( subset( RETURNS, Codename %in%
c("White","Puce","Lemon","Cyan","Carmine","Amethyst") ) )

AOV <- aov(ReturnRate ~ Codename, data=SIX)
summary(AOV)
##               Df Sum Sq Mean Sq F value Pr(>F)
## Codename       5  0.648 0.12956    38.6 <2e-16 ***
## Residuals   3205 10.757 0.00336
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

TUKEY <- TukeyHSD(AOV)
TUKEY
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = ReturnRate ~ Codename, data = SIX)
##
## $Codename
##                           diff          lwr        upr     p adj
## Carmine-Amethyst 0.0271364908  0.012684986 0.04158800 0.0000014
## Cyan-Amethyst    0.0276718319  0.012323474 0.04302019 0.0000043
## Lemon-Amethyst   0.0282167960  0.019858929 0.03657466 0.0000000
## Puce-Amethyst    0.0397759376  0.031045419 0.04850646 0.0000000
## White-Amethyst   0.0481993843  0.033472900 0.06292587 0.0000000
## Cyan-Carmine     0.0005353410 -0.018164030 0.01923471 0.9999995
## Lemon-Carmine    0.0010803052 -0.012482450 0.01464306 0.9999177
## Puce-Carmine     0.0126394468 -0.001156072 0.02643497 0.0944825
## White-Carmine    0.0210628935  0.002870485 0.03925530 0.0124797
## Lemon-Cyan       0.0005449642 -0.013969664 0.01505959 0.9999980
## Puce-Cyan        0.0121041058 -0.002628254 0.02683647 0.1773473
## White-Cyan       0.0205275525  0.001614863 0.03944024 0.0243100
## Puce-Lemon       0.0115591416  0.004395044 0.01872324 0.0000641
## White-Lemon      0.0199825883  0.006127204 0.03383797 0.0005726
## White-Puce       0.0084234467 -0.005659866 0.02250676 0.5281712

multcompLetters4(AOV,TUKEY)
## $Codename
##    White     Puce    Lemon     Cyan  Carmine Amethyst
##      "a"     "ab"      "c"     "bc"     "bc"      "d"
```

---

## Question 4: Bootstrapping and NFL statistics

The bootstrap is a powerful tool that allows us to produce a confidence interval for *almost any* parameter of the probability distribution that's responsible for generating our data, often without even knowing the identity of this distribution.

Consider the `NBA` dataset in `BAS471Sp23Unit9Parts4-5.RData`. This data includes statistics on players in the NBA from 2018 to 2022 (each row represents the statistics for the entirety of a season for one player). If you're interested, see `https://www.kaggle.com/datasets/justinas/nba-players-data` for the original source. A partial data dictionary follows:

- `player_height` - Height of the player (in centimeters)
- `player_weight` - Weight of the player (in kilograms)
- `college` - Name of the college the player attended
- `country` - Name of the country the player was born in (not necessarily the nationality)
- `draft_year` - The year the player was drafted
- `draft_round` - The draft round the player was picked
- `draft_number` - The number at which the player was picked in his draft round
- `gp` - Games played throughout the season
- `pts` - Average number of points scored across all games they played during the season
- `reb` - Average number of rebounds grabbed
- `ast` - Average number of assists distributed
- `net_rating` - Team's point differential per 100 possessions while the player is on the court
- `oreb_pct` - Percentage of available offensive rebounds the player grabbed while he was on the floor
- `dreb_ct` - Percentage of available defensive rebounds the player grabbed while he was on the floor
- `usg_pct` - Percentage of team plays used by the player while he was on the floor (FGA + Possession Ending FTA + TO) / POSS)
- `ts_pct` - Measure of the player's shooting efficiency that takes into account free throws, 2 and 3 point shots (`PTS / (2*(FGA + 0.44 * FTA))`)

- `ast_pct` - Percentage of teammate field goals the player assisted while he was on the floor

a) The "classic" procedure to create confidence intervals ("best guess" plus or minus "about two" standard errors) works great for many parameters of probability distribution under most conditions. However, there are instances when the "classic" procedure is "unavailable" and *cannot* be used, thus forcing us to bootstrap. Obviously, if the statistical theory isn't there to provide us with a value of the standard error, we must bootstrap. In what other situation is the "classic" procedure unavailable, leaving the bootstrap as our only option?

**Response: The bootstrap does not assume normality in the sampling distribution. This makes the bootstrap procedure more flexible than the "classic" procedure.**

b) The values in `pts` give the average number of points scored per game by the player over the course of a season. The histogram of values shows a right-skew, so if we wanted to summarize the distribution with a "typical" value, the median is preferred over the mean (the mean provides a better summary for symmetric distributions).

Implement the bootstrap via the `boot` command and find 95% confidence intervals for the median value of whatever probability distribution is responsible for making the values in `pts`. Report both percentile and BCa confidence intervals. Use 2499 bootstrap samples and have `set.seed(2023);` on the same line and immediately preceding the `boot` command. Note: this code will take a while to run (which means knitting will take some time).

The median value of `pts` in the data is 7.050 (`median(NBA$pts)`). Based on the 95% confidence intervals, is 7.3 a plausible value for the "true" median of the probability distribution generating the data? Why or why not?

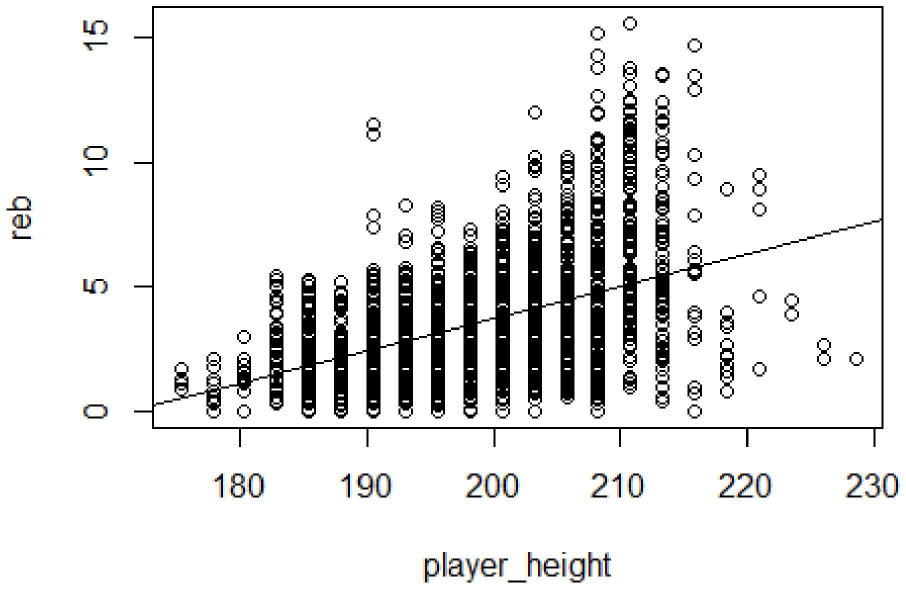
**Response: Yes, a median of 7.3 pts scored is a plasuble value for the "true" median. We know this, because 7.3 is contained in both our percentile and BCa bootstrap confidence intervals.**

```
library(boot)

set.seed(2023)

median_boot <- function(NBA, index){
  median(NBA[index])
}

boot_pts <- boot(NBA$pts, median_boot, R= 2499) ; set.seed(2023)

boot.conf.in <- boot.ci(boot_pts, type= c("perc","bca"),conf=0.95)
boot.conf.in
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2499 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_pts, conf = 0.95, type = c("perc", "bca"))
```

```
##
## Intervals :
## Level      Percentile            BCa
## 95%   ( 6.8,   7.4 )   ( 6.8,   7.4 )
## Calculations and Intervals on Original Scale
```

c) Let’s program our own (percentile) bootstrap. It would make sense if a player’s height (`player_height`) is correlated with the number of rebounds they grab (`reb`).

```
plot(reb ~ player_height, data=NBA); abline(lm(reb ~ player_height,
data=NBA))
```



```
cor(NBA$reb,NBA$player_height)
## [1] 0.4516713
(0.4516713)^2
## [1] 0.204007
```

Thinking back to BAS 320, correlations range between -1 and 1, so we would say these quantities are “fairly positively correlated”. Players with above-average heights tend to grab an above-average number of rebounds (makes sense). One way to interpret the correlation is to look at its squared value. Squaring the correlation, we find that about 20% of the variation in the number of rebounds grabbed can be attributed to differences in player height. In other words, knowing the player’s height “gets us” about 20% of the way there to knowing the number of rebounds grabbed for the season!

However, the “true” correlation between these two quantities remains unknown since our data only provides a “good guess” (we assume the pairs of observations are being generated by some joint probability distribution).

Using a `for` loop to make 9999 bootstrap samples, provide a 95% (percentile) bootstrap confidence interval for the “true” correlation. You’ll need to sample *rows* of the `NBA` dataframe and look at the correlation between the two quantities on the selected rows; the example in the notes (Unit 9C slide 42) about estimating regression coefficients provides an example of this kind of sampling.

Note: have `set.seed(2023);` on the same line and immediately preceding the `for` command. The command `cor.test` uses statistical theory to generate a confidence interval and can be used as a sanity check (your upper/lower limits should be similar, but they won’t be identical).

```
set.seed(2023)
n <- nrow(NBA)
corr_boot <- rep(NA, 9999)
for (i in 1:9999) {
  boot_idx <- sample(1:n, replace = TRUE)
  boot_data <- NBA[boot_idx, ]
  corr_boot[i] <- cor(boot_data$player_height, boot_data$reb)
}
corr_boot_ci <- quantile(corr_boot, c(0.025, 0.975))

cor.test(NBA$player_height,NBA$reb)
##
##  Pearson's product-moment correlation
##
## data:  NBA$player_height and NBA$reb
## t = 23.756, df = 2202, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4177975 0.4842910
## sample estimates:
##       cor
## 0.4516713
```

d) Using either the `boot` command or by programming your own bootstrap, find a 95% confidence interval for the 75th percentile of whatever probability distribution is generating the values contained in `NBA$ts_pct` (player’s shooting efficiency). Use 2499 bootstrap samples.

If you use `boot`, have `set.seed(2023);` on the same line and immediately preceding the `boot` command (this code will take a while to run, so knitting will take a while too). If you program your own, have `set.seed(2023);` on the same line and immediately preceding the `for` command.

```
library(boot)
set.seed(2023)
boot_75pct <- function(data, index) {
  quantile(data[index], 0.75)
}
boot_ts_pct <- boot(NBA$ts_pct, boot_75pct, R = 2499)
boot.conf.in <- boot.ci(boot_ts_pct, type = c("perc"), conf = 0.95)
boot.conf.in
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2499 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_ts_pct, conf = 0.95, type = c("perc"))
##
## Intervals :
## Level     Percentile
## 95%   ( 0.588,  0.595 )
## Calculations and Intervals on Original Scale
```