STAT S4206/S5206 Midterm

Maxime Grossman (UNI: mmg2240)

May 23, 2021

Part I: Instructions

The STAT S4206/S5206 midterm is open notes, open book(s), open computer and online resources are allowed. Students are **not** allowed to communicate with any other people regarding the exam with the exception of the instructor (Gabriel Young) and the TA (Andrew Davison). This includes emailing fellow students, using WeChat and other similar forms of communication. If there is any suspicion of one or more students cheating, further investigation will take place. If students do not follow the guidelines, they will receive a zero on the exam and potentially face more severe consequences. The exam will be posted on Friday, 05/21/2021 at 11:00AM (ET). Students are required to submit both the .pdf (or .html) and .rmd files on Canvas by Sunday, 05/23/2021 at 12:00PM (ET). Students will be given the whole 49 hour time frame to complete and upload their knitted file.

A few more recommendations follow:

- Don't forget to submit both the correct .rmd file and at least one of your .html or .pdf files.
- Save your .rmd regularly to avoid any problems if your computer crashes.
- Please ensure your output is tidy. Do not print pages and pages of data. Doing so will result in points deducted.
- Please stop working on the exam at least 30 min before it's deadline to make sure your RMarkdown file knitts properly.
- If you have a question, please include both the instructor and TA in the email thread. Don't forget that both Gabriel and Andrew are in the US so expect a delay in their response times if you are in a different timezone.

Part I: Warm-up

Run the following code:

```
setwd("C:/Users/Maxime/Desktop/Statistical Computing & Intro to Data Science/Midterm")

warm_up <- read.csv("Warm_up.csv")
head(warm_up,4)
```

```
##           V1          V2         V3         V4         V5         V6
## 1         NA  0.46620043 -1.8923489         NA  0.6408445 -1.7308123
## 2  0.1848492  0.95466641  1.2928042 -0.9007918 -1.6013778 -0.4983996
## 3  1.5878453 -0.94720635 -0.6182543  0.1499151 -0.7778154 -0.3540769
## 4 -1.1303757  0.03856309  1.0409383  1.1264128 -1.6473925         NA
##            V7           V8          V9        V10
## 1          NA  1.023567236 -0.02165721  0.7705831
## 2          NA -0.003508363  0.98448042 -0.7702532
## 3  0.04548124 -1.274958744 -0.68091825 -1.3121963
## 4 -0.46466759  1.366271568 -0.26052233 -0.4345667
```

```
dim(warm_up)
```

```
## [1] 1000   10
```

Question 1: (10 pts)

Perform the same task as the following chunk without using a loop and in one line of code.

```
my_out <- NULL
for (j in 1:10) {
  keep_vec <- NULL
  for (i in 1:1000) {
    if (!is.na(warm_up[i,j])) {
      keep_vec <- c(keep_vec,i)
    }
  }
  keep_column <- warm_up[keep_vec,j]
  my_out[j] <-  round(mean(keep_column),4)
  }
my_out
```

```
##  [1]  0.0647  0.0191  0.0693  0.0342  0.0367  0.0012 -0.0783  0.0333 -0.0723
## [10]  0.0409
```

The solution is as follows:

```
round(apply(warm_up, 2, mean, na.rm=TRUE), digits=4)
```

```
##      V1      V2      V3      V4      V5      V6      V7      V8      V9     V10
##  0.0647  0.0191  0.0693  0.0342  0.0367  0.0012 -0.0783  0.0333 -0.0723  0.0409
```

Part II: Data Cleaning and Graphics

Run the following code:

```
data <- read.csv("PlayerProfiles.csv")
names(data) <- c("Name","Country","Age","DOB","Nickname","PDC_Ranking","Tour_Card","Career_Earnings")
head(data,4)
```

```
##                 Name     Country Age       DOB    Nickname PDC_Ranking
## 1 Michael van Gerwen Netherlands  32 4/25/1989 Mighty Mike           1
## 2       Peter Wright    Scotland  51 3/10/1970  Snake Bite           2
## 3       Gerwyn Price       Wales  36  3/7/1985  The Iceman           3
## 4       Adrian Lewis     England  36 1/21/1985     Jackpot          13
##   Tour_Card Career_Earnings
## 1       Yes     Â£8,321,167
## 2       Yes     Â£3,469,888
## 3       Yes     Â£1,497,803
```

```
## 4       Yes     Â£3,137,634
```

```
dim(data)
```

```
## [1] 96  8
```

Question 2: (5 pts)

There appears to be some repeated cases in the dataframe (**data**). Identify the number of repeated cases in this dataset. There are many ways to solve this problem, e.g., you can look at repeated names or nicknames and then figure out how to extract this information from the dataframe. I personally used a loop for this problem.

```
level <- levels(data$Name)          # get levels of Name in alphabetical order
rows <- length(level)               # count how many levels

w <- NULL                           # create empty vector to store which( ) results
vectorofduplicates <- NULL          # create empty vector to store duplicate names
count <- 0                          # set index
j <- 1                              # set index
c <- 0                              # set index


for(i in 1:rows){
  w <- (which(data$Name == level[i]))    # of each level, which rows match it?
  if(length(w)>1){                       # if more than 1 row matches, enter if condition
    vectorofduplicates[j]<-level[i]      # store the player name in vectorofduplicates
    count <- count+1                     # keep count of duplicate number
    j <- j+1                             # increase index for next time
    }
  }


vectorofduplicates                      # list of reoccuring names
```

```
##  [1] "Brendan Dolan"         "Gerwyn Price"          "Jelle Klaasen"
##  [4] "John Henderson"        "Kim Huybrechts"        "Kirk Shepherd"
##  [7] "Mensur Suljovic"       "Michael van Gerwen"    "Peter Wright"
## [10] "Ricky Evans"           "Robert Thornton"       "Ronny Huybrechts"
## [13] "Ross Smith"            "Scott Waites"          "Simon Whitlock"
## [16] "Vincent van der Voort"
```

```
count                                   # number of unique names duplicated
```

```
## [1] 16
```

```
dupes <- 0                              # we check for how many duplicates exist
for(k in vectorofduplicates){
  dupes <- dupes + sum(data$Name == k)
  }; dupes
```

```
## [1] 34
```

```
# table(data$Name)
```

If we ask how many unique names appear more than once, the answer is *count = 16*.

We found that duplicates or *dupes = 34*. In other words, when viewing all observations, in 34 instances we can deduce that there is an identical observation in another row besides the one currently under observation.

The results can be reconciled by the *table(data$Name)* function, where we can count 14 names which appear twice and two names which appear three times, for a total count of 34. This function has been commented out because it is only used as a manual check.

Question 3: (5 pts)

Create a new dataframe called **new_data** that excludes all repeats. For partial credit, you can manually identify the repeats and remove these cases with a basic subsetting command. Display the head and dimension of **new_data**. I personally used a loop for this problem.

```
deleterows <- NULL              # create a vector to catalog rows to delete
d <- 1                          # set index

for(l in 1:count){

  wrow <- which(data$Name == vectorofduplicates[l])    # which rows are duplicate names?
  deleterows[d] <- wrow[2]                             # store 2nd matching row in vector
  deleterows[d+1] <- wrow[3]                           # store 3rd matching row if applicable
  d <- d+2                                             # increase index

  }

deleterows <- deleterows[!is.na(deleterows)]    # remove any NAs

new_data <- data[-deleterows,]                  # delete rows from data
head(new_data)
```

```
##                 Name     Country Age       DOB    Nickname PDC_Ranking
## 1 Michael van Gerwen Netherlands  32 4/25/1989 Mighty Mike           1
## 2       Peter Wright    Scotland  51 3/10/1970  Snake Bite           2
## 3       Gerwyn Price       Wales  36  3/7/1985  The Iceman           3
## 4       Adrian Lewis     England  36 1/21/1985     Jackpot          13
## 5         James Wade     England  38  4/6/1983 The Machine           9
## 6      Andy Hamilton     England  54 3/16/1967  The Hammer         105
##   Tour_Card Career_Earnings
## 1       Yes     Â£8,321,167
## 2       Yes     Â£3,469,888
## 3       Yes     Â£1,497,803
## 4       Yes     Â£3,137,634
## 5       Yes     Â£3,458,893
## 6    2 Year     Â£1,089,788
```

```
dim(new_data)
```

```
## [1] 78  8
```

```
# A (much) easier alternative:
# library(dplyr)
# new_data <- distinct(data)
```

Our strategy was as follows:

Find which rows we need to delete using *which( )* function to see if an observation shows up in more than one place

We keep the first *which( )* match in place but store the values of the locations of the second and third matches in a vector; we delete these rows later on

Although we could have used one single function, *distinct(data)*, to accomplish this task, I employed a more "manual" method in an effort to demonstrate concepts we've learned in class.

Question 4: (10 pts)

Use base R character string functions to convert the **Career_Earnings** variable into a numeric mode. For example, the symbol £535,131 should be converted to 535132. Your converted variable should be appended to your dataframe and named **Career_Earnings_NUM**. You can use the original data or **new_data** to solve this problem. Display the head of the numeric vector **Career_Earnings_NUM**.

```
Career_Earnings_NUM <- substr(new_data$Career_Earnings, start=3, stop=20)

careervalues <- strsplit(Career_Earnings_NUM, split = ",")

for(o in 1:rows){
  Career_Earnings_NUM[o] <- paste(careervalues[[o]], collapse="")
}

Career_Earnings_NUM <- as.numeric(Career_Earnings_NUM)

new_data <- data.frame(new_data, Career_Earnings_NUM)

head(new_data$Career_Earnings_NUM)
```

```
## [1] 8321167 3469888 1497803 3137634 3458893 1089788
```

Our strategy was as follows:

1. **Remove *char* symbols in front of each entry using *substr( )***
2. **Use *strsplit( )* to remove commas then use *paste( )* and collapse**
3. **Coerce as *as.numeric( )* then add back to dataframe**

Question 5: (10 pts)

Using relevant R functions, identify the 5 most frequent countries measured in this dataset. Display the country and its frequency. Note that England should have the highest frequency. You can use the original data or **new_data** to solve this problem. Also create a barplot displaying the 5 most frequent countries.

```
# we can first use the table( ) function and sort by decreasing

sort(table(new_data$Country), decreasing=TRUE)
```

```
##
##         England      Netherlands         Scotland            Wales
##              39               10                4                4
```

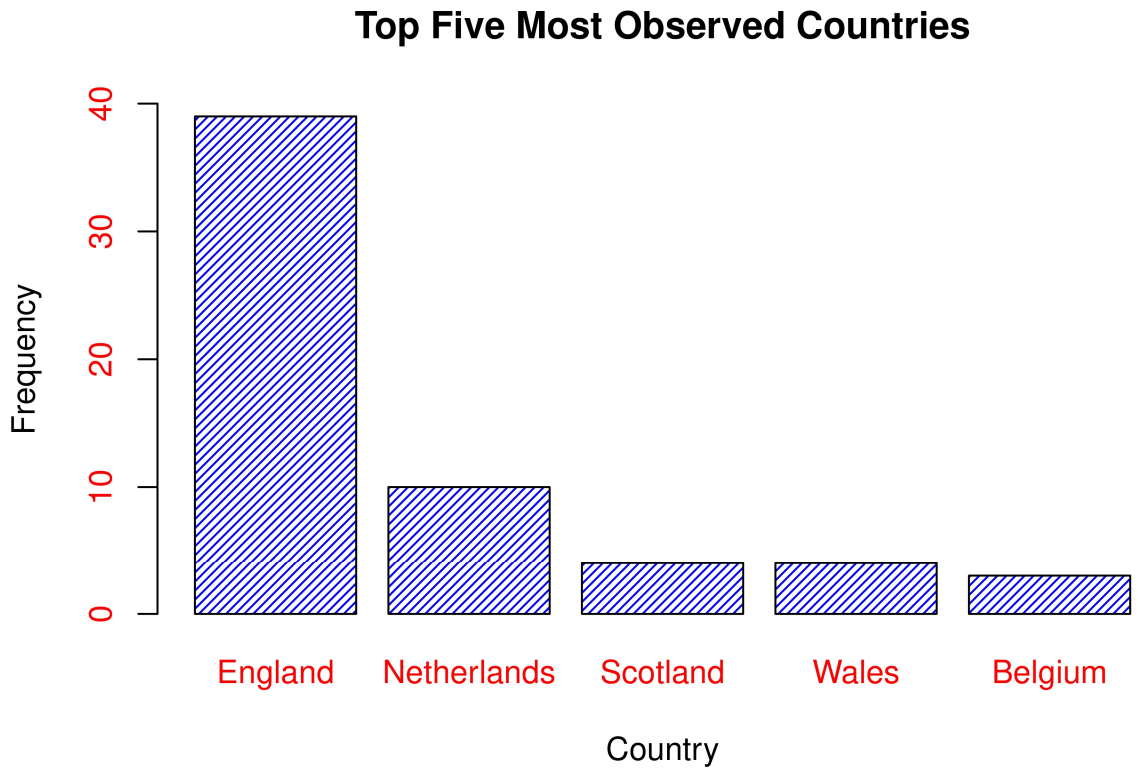
```
##          Belgium Northern Ireland        Australia          Austria
##                3                3                2                2
##          Germany            Spain           Brazil           Canada
##                2                2                1                1
##           Greece          Ireland           Latvia        Lithuania
##                1                1                1                1
##     South Africa
##                1
```

```
# filter to the top five in the index

top5countries <- sort(table(new_data$Country), decreasing=TRUE)[1:5]; top5countries
```

```
##
##     England Netherlands    Scotland       Wales     Belgium
##          39          10           4           4           3
```

```
barplot(top5countries, main="Top Five Most Observed Countries",
        xlab = "Country",
        ylab = "Frequency",
        col="blue",
        border="black",
        ylim=c(0,40),
        density=30,
        angle=45,
        col.axis="red",
        cex.axis=1)
```



There are actually two countries which tied for fifth most frequent, but I decided to only show five countries and not six for simplicity. This can be verified by observing the *table( )* function results. Belgium and North Ireland both appear three times and so are tied for fifth place. However, I took the sorted table's index from 1 to 5 and applied the *barplot( )* function, which only took Belgium most likely as a matter of alphabetical chronology. We could have easily included North Ireland by adjusting the index from 1 to 6.

Question 6: (5 pts)

Create a new variable in the dataframe named **England**, which reads "England" if the case belongs to England and "NotEngland" otherwise. You can use the original data or **new_data** to solve this problem. Display the head of the **England** variable.

```
new_data$England <- ifelse(new_data$Country=="England", "England", "NotEngland")
head(new_data$England)
```

```
## [1] "NotEngland" "NotEngland" "NotEngland" "England"    "England"
## [6] "England"
```

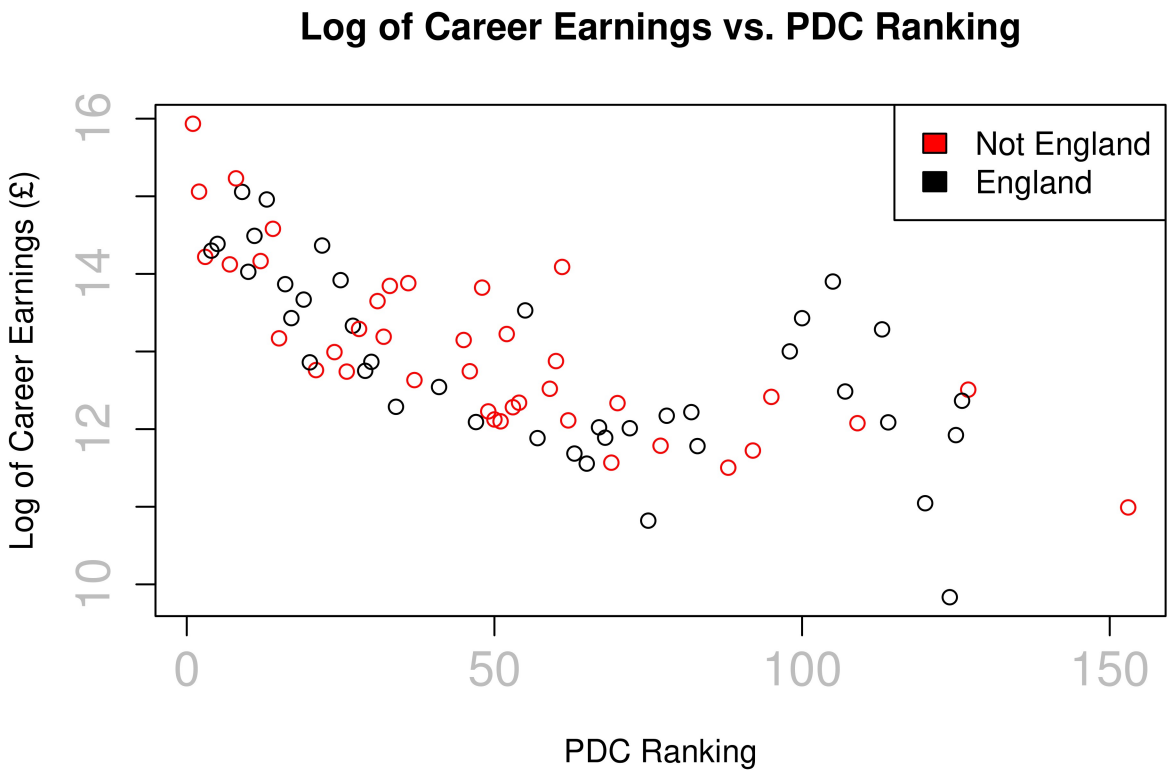
Question 7: (5 pts)

Using Base R, construct a scatter plot of **log(Career_Earnings_NUM)** versus **PDC_Ranking**, split by **England**. Make sure to include a legend and label your axes appropriately.

If you could not solve question 4, construct a base R scatter plot of **PDC_Ranking** versus **Age**, split by **England**. Make sure to include a legend and label your axes appropriately.

```
plot(new_data$PDC_Ranking, log(new_data$Career_Earnings_NUM),
     main="Log of Career Earnings vs. PDC Ranking",
     xlab = "PDC Ranking", ylab = "Log of Career Earnings (£)",
     col=factor(new_data$England),
     col.axis="grey",
     cex.axis=1.5)

legend("topright", legend = c("Not England", "England"),
       fill= unique(factor(new_data$England)))
```



The scatterplot tells us that higher ranking individuals enjoy higher earnings. This is intuitive.

Question 8: (5 pts)

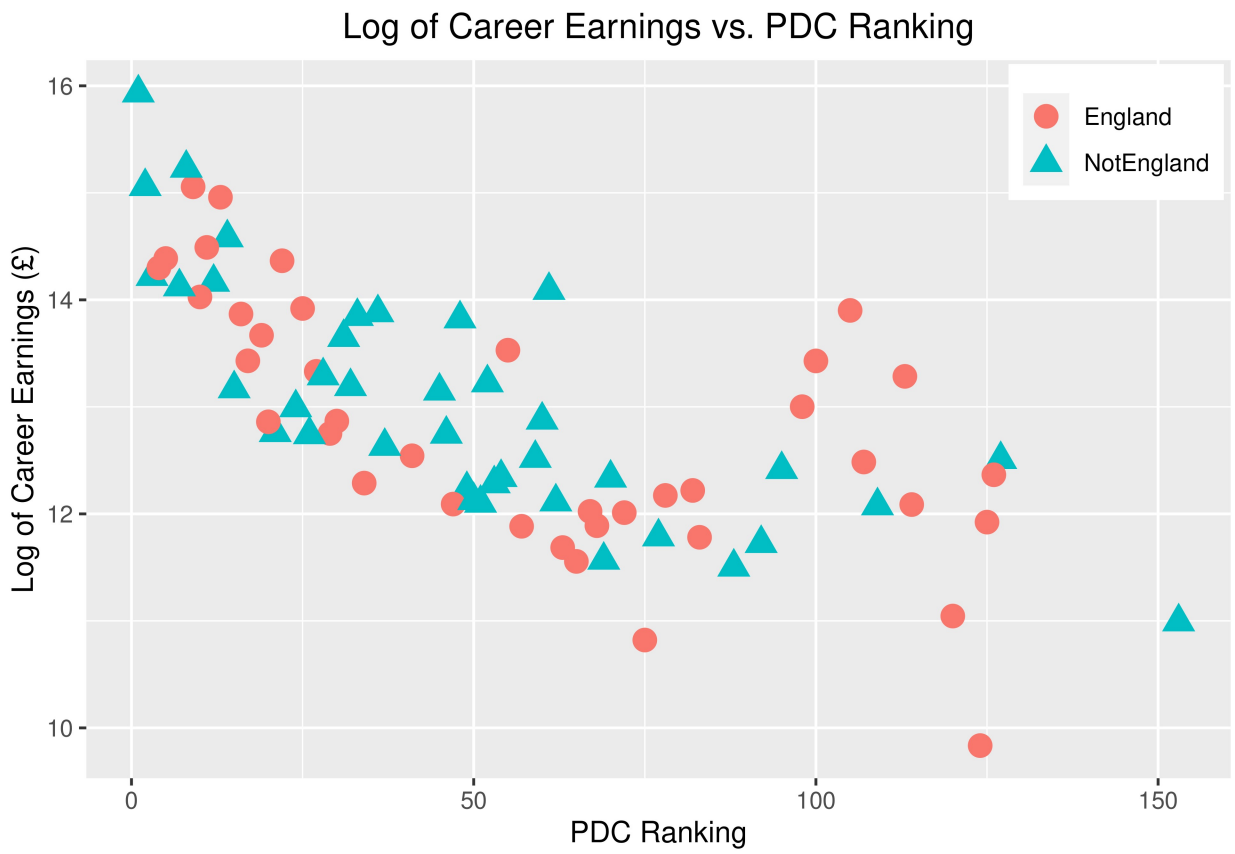
Using ggplot, construct a scatter plot of **log(Career_Earnings_NUM)** versus **PDC_Ranking**, split by **England**. Make sure to include a legend and label your axes appropriately.

If you could not solve question 4, construct a ggplot scatter plot of **PDC_Ranking** versus **Age**, split by **England**. Make sure to include a legend and label your axes appropriately.

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
ggplot(data=new_data) +
  geom_point(mapping = aes(x=PDC_Ranking,
       y=log(Career_Earnings_NUM),
       color=England,
       shape=England),
       size=4) +
  labs(title = "Log of Career Earnings vs. PDC Ranking",
       x = "PDC Ranking",
       y = "Log of Career Earnings (£)")  +
  theme(plot.title = element_text(hjust=0.5)) +
  theme(legend.title = element_blank()) +
   theme(legend.position = c(.9, .9))
```



Part II: Nonparametric Procedures

Question 9: (10 pts)

Compute the the two conditional probabilities (or proportions):

I) Given the respondent represents England, what is the probability that their **PDC_Ranking** is in the upper 25%?

$$p_1 = P(\text{High Rank}|\text{England})$$

II) Given the respondent does not represent England, what is the probability that their **PDC_Ranking** is in the upper 25%?

$$p_2 = P(\text{High Rank}|\text{Not England})$$

I)

```
quartile <- quantile(new_data$PDC_Ranking, probs = 0.25)

# count if an entry is in the upper 25 percentile
```

```r
isupper25 <- sum(ifelse(new_data$PDC_Ranking >= quartile, 1, 0))

# count if an entry is in the upper 25 percentile and is England

isenglandandupper25 <- sum(ifelse(new_data$PDC_Ranking >= quartile & new_data$England == "England", 1, 0

# count if an entry is England

isengland <- sum(ifelse(new_data$England == "England", 1, 0))

p1 <- isenglandandupper25/isengland; p1
```

```
## [1] 0.7179487
```

We use

$$P(High\ Rank|England) = \frac{P(England|High\ Rank)P(High\ Rank)}{P(England)} =$$

$$= \frac{P(England\ and\ High\ Rank)}{P(High\ Rank)} \frac{P(High\ Rank)}{P(England)} = \frac{P(England\ and\ High\ Rank)}{P(England)}$$

The probability that a respondent is of High Rank given they are from England is: 0.7179487

II)

```r
# count if not England

notengland <- sum(ifelse(new_data$England == "NotEngland", 1, 0))

# count if not England and in the upper 25 percentile

notenglandandupper25 <- sum(ifelse(new_data$PDC_Ranking >= quartile &
                                   new_data$England == "NotEngland", 1, 0))

p2 <- notenglandandupper25/notengland; p2
```

```
## [1] 0.7692308
```

We use

$$P(High\ Rank|Not\ England) = \frac{P(Not\ England|High\ Rank)P(High\ Rank)}{P(Not\ England)} =$$

$$= \frac{P(Not\ England\ and\ High\ Rank)}{P(High\ Rank)} \frac{P(High\ Rank)}{P(Not\ England)} = \frac{P(Not\ England\ and\ High\ Rank)}{P(Not\ England)}$$

The probability that a respondent is of High Rank given they are not from England is: 0.7692308

Question 10: (20 pts)

Consider testing the the null alternative pair:

$$H_0 : p_1 - p_2 = 0 \quad \text{versus} \quad H_A : p_1 - p_2 \neq 0$$

Run a bootstrap procedure to test the above hypothesis. To accomplish this task, construct a 95% bootstrap interval on the parameter $\theta = p_1 - p_2$ and check if 0 falls in the interval.

Note for comparison, you can look up the two-sample proportions z-test, which is the analogous parametric procedure. The final results should be similar. Also note that the bootstrap hypothesis test introduced above is two-tailed, which only shows that the proportions are statistically different. A one tailed test might be more appropriate but this approach is not required for the midterm.

```
theta <- p1 - p2

B <- 1000                # we will create 1000 datasets through bootstrapping
p1p2estimates <- 1000    # from bootstrapping, we create 1000 estimates of p1 and p2

# estimates will store the resampled PDC_Ranking and England data

estimates <- matrix(NA, nrow=B, ncol=2)

# we will be storing char and numeric so coerce estimates into a df

estimates <- data.frame(estimates)
colnames(estimates) <-  c("PDC_Ranking", "England")

# p1p2hat will store the p1 and p2 estimates

p1p2hat <- matrix(NA, nrow=p1p2estimates, ncol=2)
colnames(p1p2hat) <-  c("p1 boot", "p2 boot")

for (f in 1:p1p2estimates) {
  for (e in 1:B){
    randomrow <- sample(1:78, 1, replace=T)
    estimates[e,1] <- as.numeric(new_data$PDC_Ranking[randomrow])
    estimates[e,2] <- new_data$England[randomrow]
    }

  # find quantile

  Q <- quantile(estimates[,1], probs=0.25)

  # count of all high ranking

  H <- sum(ifelse(estimates["PDC_Ranking"]>=Q, 1, 0))

  # count of all England

  E <- sum(ifelse(estimates["England"]=="England",1,0))
```

```
  # count of all not England

  NE <- sum(ifelse(estimates["England"]=="NotEngland",1,0))

  # count of all high rank and England

  HE <-  sum(ifelse(estimates["PDC_Ranking"]>=Q & estimates[,2]=="England", 1, 0))

  # count of all high rank and not England

  HNE <- sum(ifelse(estimates["PDC_Ranking"]>=Q & estimates[,2]=="NotEngland", 1, 0))

  p1sim <- HE/E
  p2sim <- HNE/NE

  p1p2hat[f,1] <- p1sim
  p1p2hat[f,2] <- p2sim

}

head(p1p2hat)
```

```
##        p1 boot   p2 boot
## [1,] 0.7652672 0.7394958
## [2,] 0.7378049 0.7637795
## [3,] 0.7336066 0.7753906
## [4,] 0.7346939 0.7852495
## [5,] 0.7272727 0.7782178
## [6,] 0.7151394 0.7951807
```

```
theta_hat <- p1p2hat[,1] - p1p2hat[,2]

p1p2hat <- data.frame(p1p2hat, theta_hat)

q975 <- quantile(p1p2hat[,3], probs=0.975)
q025 <- quantile(p1p2hat[,3], probs=0.025)

L <- 2*theta - q975
U <- 2*theta - q025

boot_interval <- c(L,U); boot_interval
```

```
##        97.5%         2.5%
## -0.121478671 -0.007729611
```

Below are some examples of bootstrap confidence intervals:

1. $(-0.126318492, -0.006725964)$

2. $(-0.128256136, -0.006465649)$

3. $(-0.121891986, -0.003564977)$

4. $(-0.122377116, -0.001343309)$

5. $(-0.126446281, -0.006452742)$

After multiple pulls leading to the same result, we can conclude that we should reject the null hypothesis as our null hypothesis value of

$$\theta = \theta_0 = 0$$

does *not* fall in the interval. Thus we accept the alternative hypothesis.

This is expected because we almost always see $p_1 < p_2$ in our bootstrap estimates, so it would be difficult to conclude that $p_1 = p_2$.

Part III: Writing a R function

Question 11: (10 pts)

Write a function named **my_description** that creates a written description of a particular dart player. The function should have two inputs, (1) the case as an integer and (2) the data frame (**new_data** or **data**). For example, respondent 3 is Gerwyn Price, i.e., **my_description(case=3,df=new_data)**. Your function's output should read similar to:

"Gerwyn Price aka The Iceman, is a 36 year old dart player from Whales. The Iceman is currently ranked number 3 and his career earnings total 1497803 dollars."

Try to make this exercise fun. You can write the above expression however you want but your function must generalize to multiple cases. You can also include extra information if desired. To check your function, make sure to display your function's output for Gerwyn Price.

```
my_description <- function(case, df) {


  ifelse(df=="new_data", df <- new_data, df <- data)

  cat("\nName: \t\t\t\t", levels(df[,1])[df[case,1]], "\n")
  cat("Nickname: \t\t\t", levels(df[,5])[df[case,5]], "\n")
  cat("Age: \t\t\t\t", df[case,3], "\n")
  cat("Country: \t\t\t", levels(df[,2])[df[case,2]], "\n")
  cat("Ranking: \t\t\t", df[case,6], "\n")
  cat("Cake Day: \t\t\t", levels(df[,4])[df[case,4]], "\n")
  cat("Career Earnings: \t", substr(levels(df[,8])[df[case,8]], start=2, stop=12), "\n")

  }


# output for Gerwyn Price:

my_description(3, new_data)
```

```
##
## Name:                Gerwyn Price
## Nickname:            The Iceman
## Age:                 36
## Country:             Wales
## Ranking:             3
## Cake Day:            3/7/1985
## Career Earnings:     £1,497,803
```

I wrote the output in a column format for optimum readability.

Question 12: (5 pts)

Use a vectorized operation or an apply function to compute all dart player's descriptions. Show the head of your resulting vector.

```
# sapply(rownames(new_data), my_description, new_data)


for(a in 1:6){
  my_description(a, new_data)
}
```

```
##
## Name:                Michael van Gerwen
## Nickname:            Mighty Mike
## Age:                 32
## Country:             Netherlands
## Ranking:             1
## Cake Day:            4/25/1989
## Career Earnings:     £8,321,167
##
## Name:                Peter Wright
## Nickname:            Snake Bite
## Age:                 51
## Country:             Scotland
## Ranking:             2
## Cake Day:            3/10/1970
## Career Earnings:     £3,469,888
##
## Name:                Gerwyn Price
## Nickname:            The Iceman
## Age:                 36
## Country:             Wales
## Ranking:             3
## Cake Day:            3/7/1985
## Career Earnings:     £1,497,803
##
## Name:                Adrian Lewis
## Nickname:            Jackpot
## Age:                 36
## Country:             England
## Ranking:             13
```

```
## Cake Day:             1/21/1985
## Career Earnings:      £3,137,634
##
## Name:                 James Wade
## Nickname:             The Machine
## Age:                  38
## Country:              England
## Ranking:              9
## Cake Day:             4/6/1983
## Career Earnings:      £3,458,893
##
## Name:                 Andy Hamilton
## Nickname:             The Hammer
## Age:                  54
## Country:              England
## Ranking:              105
## Cake Day:             3/16/1967
## Career Earnings:      £1,089,788
```

The *sapply( )* function has been commented out as it will print out a very long list, which we are instructed not to do. Please feel free to run it to confirm it works.

I used a *for( )* loop to print out the first six entries, which gives more control than the *head( )* function.