STAT 123 - Homework 3

Parker DeBruyne - V00837207

09/03/2022

1. Download and save the homework3Data.csv dataset and read it into R. This data set contains 6 numerical variables A, B, C, X, Y, Z.

```
hw3Data = read.csv("homework3Data.csv")
head(hw3Data, 3)
```

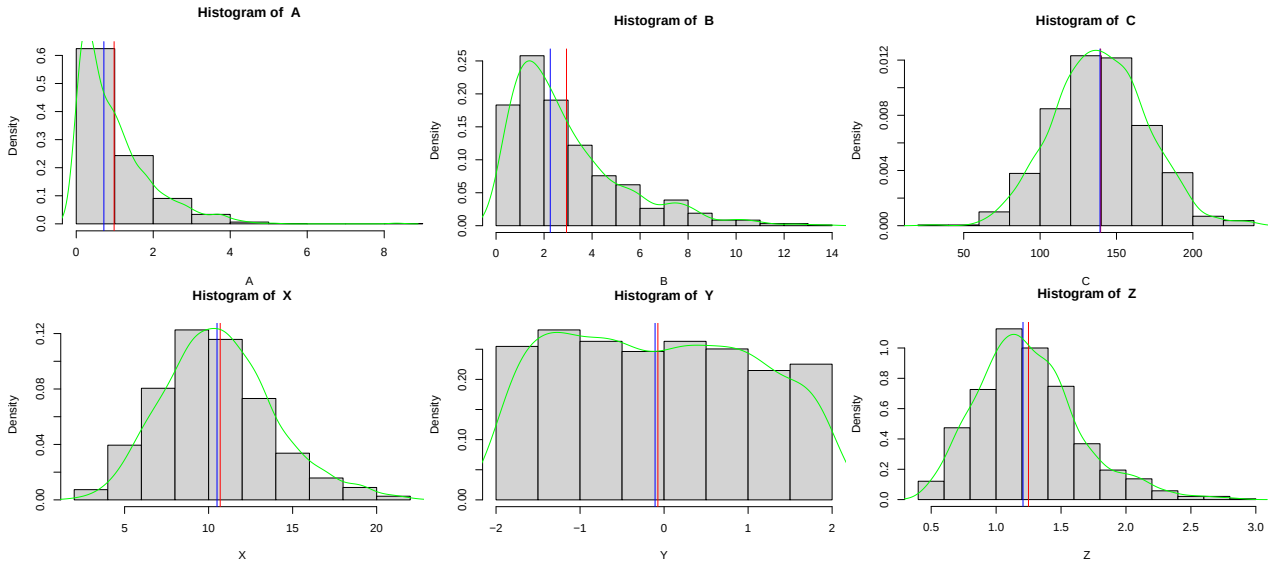
```
##   X.1         A        B        C  X          Y         Z
## 1   1 0.9102545 1.106865 124.6358  9 -0.7828927 0.9983816
## 2   2 1.0540369 2.104931 214.5555 19 -0.1010444 1.2075391
## 3   3 0.6864621 3.361089 170.2348 14  1.9741031 1.4268379
```

(a) If you use the function hist() to plot a histogram followed by the function abline(v=3,col='red'), this will add a red vertical line at x = 3.

Using these functions:

- plot a histogram for each of the variables.
- add vertical lines for the sample mean and the sample median of those variables. Make the sample mean lines red and the sample median lines blue.
- add a green density curve to each plot.
- make sure your histogram has a main title.

```
hw3Data_names = colnames(hw3Data)

for(i in 2:length(hw3Data_names)){
hist(hw3Data[[i]],
     main = paste("Histogram of ",
                  hw3Data_names[i]),
     prob=TRUE,
     xlab = hw3Data_names[i])
abline(v=mean(hw3Data[[i]]),col='red')
abline(v=median(hw3Data[[i]]),col='blue')
lines(density(hw3Data[[i]]),col='green')
}
```



(b) One of the variables is normally distributed. Determine which variable it is and justify why you think it is that variable.

Ans: Variable C is normally distributed because the mean and the median are approximately equal and overlap.

(c) For the normally distributed variable you identified in part (b), use the 68 — 95 — 99.7 rule to determine the intervals such that approximately 68% of the data, 95% of the data, and 99.7% of the data lie within those intervals.

```
sigma_C = sd(hw3Data[["C"]])
mean_C = mean(hw3Data[["C"]])

C_68 = round((mean_C + sigma_C), 2)
print(paste("68th percentile =", C_68))
```

```
## [1] "68th percentile = 169.94"
```

```
C_95 = round((mean_C + 2*sigma_C), 2)
print(paste("95th percentile =", C_95))
```

```
## [1] "95th percentile = 200.27"
```

```
C_99.7 = round((mean_C + 3*sigma_C), 2)
print(paste("99.7th percentile =", C_99.7))
```

```
## [1] "99.7th percentile = 230.59"
```

(d) Use the quantile() function to approximate those same intervals. Are the intervals the same?

```
quantile(hw3Data[["C"]], 0.68)
```

```
##      68%
## 153.7819
```

```
quantile(hw3Data[["C"]], 0.95)
```

```
##      95%
## 190.4729
```

```
quantile(hw3Data[["C"]], 0.997)
```

```
##    99.7%
## 232.3657
```

(e) Use the qnorm() function (with the sample mean and sample standard deviation) to approximate those same intervals. Are these intervals the same as the intervals in either part (c) or part (d)?

```
round(qnorm(c(0.68,0.95,0.997), mean= mean_C, sd=sigma_C), 2)
```

```
## [1] 153.80 189.50 222.94
```

Ans: They're very close, but not exactly the same!

(f) Suppose you wish to estimate the population mean for the normally distributed variable you identified in part (b). Compute the following:

- an estimate of the population mean.
- the estimated standard error of the statistic.
- the critical value for an 88% confidence interval.
- a 88% confidence interval for the population mean.

```
n = length(hw3Data)

#Estimate the population mean
boot_sample = numeric()
for(i in 1:10000){
  temp_samp = sample(hw3Data[["C"]], n, replace=TRUE)
  temp_mean = mean(temp_samp)
  boot_sample[i] = temp_mean
}

boot_mean = mean(boot_sample)
print(paste("Bootstrapped population mean =", round(boot_mean, 2)))
```

```
## [1] "Bootstrapped population mean = 139.53"
```

```
#Estimate the standard error
std_err = sd(hw3Data[["C"]])/sqrt(n)
print(paste("Standard error of C =", round(std_err,2)))
```

```
## [1] "Standard error of C = 11.46"
```

```
#Estimate the critical value for an 88% confidence interval.
diff_88 = (100-88)/2
interval_88 = c(diff_88, 100-diff_88)

crit_val_88 = qnorm(interval_88/100, mean=0, sd=1)
print(paste("Critical Value estimate for an 88% confidence interval =", round(crit_val_88[2], 2)))
```

```
## [1] "Critical Value estimate for an 88% confidence interval = 1.55"
```

```
#Estimate a 88% confidence interval for the population mean.

confidence_int_88 = c(boot_mean - crit_val_88[2]*std_err, boot_mean + crit_val_88[2]*std_err)

print(paste("Confidence interval for 88% = (",
            round(confidence_int_88[1], 2),
            ",",
            round(confidence_int_88[2], 2),
            ")"))
```

```
## [1] "Confidence interval for 88% = ( 121.71 , 157.35 )"
```

2. For this question, you will need to install the package 'dplyr' into R by typing in the command install.packages('dplyr'). Then you need to load dplyr into R by typing in the command library(dplyr). We will be using the starwars data set that is built into the dplyr package.

```
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
```

(a) Create a vector called names which contains the names of starwars characters that are included in the data set.

```
names = starwars$name
```

(b) The function nchar() determines the number of characters in a string. How many characters are in the 5th, 20th, and 34th elements of the names vector?

```
print(paste("Number of characters in the 5th element = ",nchar(names[5])))
```

```
## [1] "Number of characters in the 5th element =  11"
```

```
print(paste("Number of characters in the 20th element = ",nchar(names[20])))
```

```
## [1] "Number of characters in the 20th element =  9"
```

```
print(paste("Number of characters in the 34th element = ",nchar(names[34])))
```

```
## [1] "Number of characters in the 34th element =  13"
```

(c) Create an empty numeric vector called num_char. Write a loop which calculates the number of characters in each element of the names vector, and puts the corresponding number in the num char vector.

```
num_char = numeric()
for(i in 1:length(names)){
  num_char[i] = nchar(names[i])
}

num_char
```

```
##  [1] 14  5  5 11 11  9 18  5 17 14 16 14  9  8  6 21 14 16  4  9  9  5  5 16  5
## [26]  6 10 12 21  9 12 11 13 13 12 10  8  5  7 13 14 10 11 11  8  7 14 10 12  9
## [51]  9 10 11 11  8 10 12  5 11 17 15 13  5  5 19 10 10 15  7  7 10 13  6 10  8
## [76]  8  8  7 15  9 10  4  3 11  3 14 13
```

(d) Now do the same thing that you did in part (c) using the lapply() or sapply() function in R. Be careful that your output is a vector.

```
num_char_sapply = sapply(names, nchar)
num_char_sapply
```

```
##        Luke Skywalker                 C-3PO                R2-D2
##                    14                     5                     5
##           Darth Vader           Leia Organa             Owen Lars
##                    11                    11                     9
##    Beru Whitesun lars                 R5-D4     Biggs Darklighter
##                    18                     5                    17
##        Obi-Wan Kenobi      Anakin Skywalker        Wilhuff Tarkin
##                    14                    16                    14
##             Chewbacca              Han Solo                Greedo
##                     9                     8                     6
## Jabba Desilijic Tiure        Wedge Antilles      Jek Tono Porkins
##                    21                    14                    16
##                  Yoda             Palpatine             Boba Fett
##                     4                     9                     9
##                 IG-88                 Bossk      Lando Calrissian
##                     5                     5                    16
##                 Lobot                Ackbar           Mon Mothma
##                     5                     6                    10
##          Arvel Crynyd Wicket Systri Warrick             Nien Nunb
##                    12                    21                     9
##          Qui-Gon Jinn           Nute Gunray         Finis Valorum
##                    12                    11                    13
##         Jar Jar Binks          Roos Tarpals            Rugor Nass
##                    13                    12                    10
##              Ric Olié                 Watto              Sebulba
##                     8                     5                     7
##         Quarsh Panaka        Shmi Skywalker            Darth Maul
##                    13                    14                    10
##           Bib Fortuna           Ayla Secura             Dud Bolt
##                    11                    11                     8
##               Gasgano        Ben Quadinaros            Mace Windu
##                     7                    14                    10
```

```
##          Ki-Adi-Mundi             Kit Fisto             Eeth Koth
##                    12                     9                     9
##            Adi Gallia          Saesee Tiin          Yarael Poof
##                    10                    11                    11
##              Plo Koon           Mas Amedda         Gregar Typho
##                     8                    10                    12
##                 Cordé          Cliegg Lars    Poggle the Lesser
##                     5                    11                    17
##       Luminara Unduli         Barriss Offee                 Dormé
##                    15                    13                     5
##                Dooku   Bail Prestor Organa           Jango Fett
##                     5                    19                    10
##            Zam Wesell       Dexter Jettster              Lama Su
##                    10                    15                     7
##               Taun We            Jocasta Nu        Ratts Tyerell
##                     7                    10                    13
##               R4-P17           Wat Tambor             San Hill
##                     6                    10                     8
##              Shaak Ti             Grievous              Tarfful
##                     8                     8                     7
##       Raymus Antilles            Sly Moore           Tion Medon
##                    15                     9                    10
##                 Finn                  Rey          Poe Dameron
##                     4                     3                    11
##                  BB8       Captain Phasma        Padmé Amidala
##                     3                    14                    13
```

3. Consider again the homework3Data.csv dataset and the variable X.

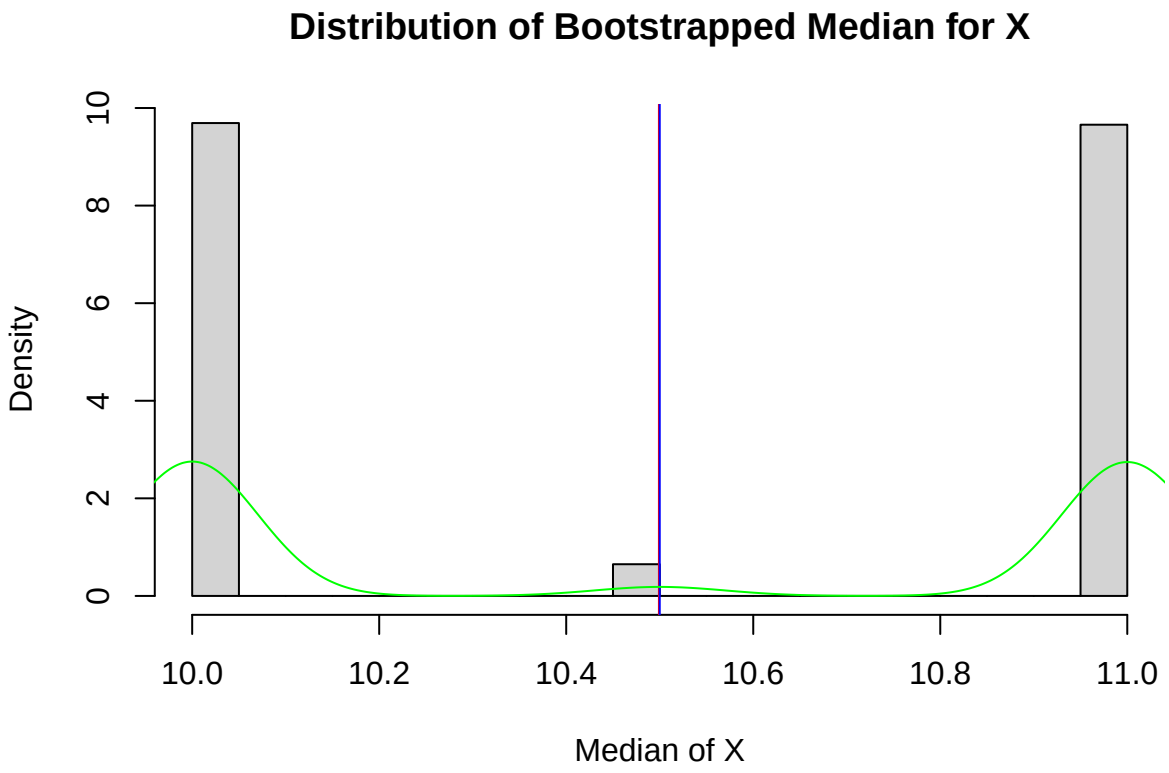
(a) Write a bootstrap computing the median on 10, 000 samples (with replacement) of size 600 of the variable X.

```
n = length(hw3Data[["X"]])

boot_sample = numeric()
for(i in 1:10000){
  temp_samp = sample(hw3Data[["X"]], 600, replace=TRUE)
  temp_median = median(temp_samp)
  boot_sample[i] = temp_median
}

boot_median = mean(boot_sample)
```

(b) Plot the resulting sampling distribution for the median of X.

```
hist(boot_sample,
     main="Distribution of Bootstrapped Median for X",
     xlab="Median of X",
     prob=TRUE)
abline(v=mean(boot_sample),col='red')
abline(v=median(boot_sample),col='blue')
```

```
lines(density(boot_sample),col='green')
```



(c) Determine an estimate for the median of X.

```
boot_median = mean(boot_sample)
print(paste("Estimated median for the population of X =", round(boot_median, 2)))
```

```
## [1] "Estimated median for the population of X = 10.5"
```

(d) Compute a 95% confidence interval for the median of X.

```
#Estimate the standard error
std_err = sd(boot_sample)/sqrt(n)

#Estimate the critical value for an 95% confidence interval.
diff_95 = (100-95)/2
interval_95 = c(diff_95, 100-diff_95)
crit_val_95 = qnorm(interval_95/100, mean=0, sd=1)

#Estimate a 95% confidence interval for the population median.
confidence_int_95 = c(boot_median - crit_val_88[2]*std_err, boot_median + crit_val_88[2]*std_err)

print(paste("Confidence interval for 95% = (",
            round(confidence_int_95[1], 2),
            ",",
            round(confidence_int_95[2], 2),
            ")"))
```

```
## [1] "Confidence interval for 95% = ( 10.47 , 10.52 )"
```