UW STATR 501A - Homework #4

Melissa Gaughan

2022-10-27

# Instructions

Please submit a single zipped folder with all associated files [text report (.pdf, .docx, .txt), R code (.R, .Rmd), data files (.csv, .xlsx, .txt)]. Upload the completed homework assignment into the Canvas drop-box.

Probability problems may be unintuitive - hence the emphasis on simulation. Please discuss the problems amongst yourselves on the forum, but make sure to submit your own work (code and text).

## Problem #1 - Reading

### Part a

Read the file lab4\_corrected.html (under modules of the course site)

### Part b

Read Ch. 1 Probability Models and Ch. 2 Random Variables and Distributions in Probability and Statistics.

### Part c

Read Ch. 8 Probability Distributions in An Introduction to R.

## Problem #2 - Random Variables I

### Part a

Simulate 1000 numbers from the [standard uniform distribution](https://en.wikipedia.org/wiki/Continuous_uniform_distribution) , and report the [sample standard deviation](https://en.wikipedia.org/wiki/Standard_deviation#Sample_standard_deviation) () of these numbers.

#<Write code here>  
  
sample\_1000 <- runif(1000)  
  
standard\_deviation <- sqrt(var(sample\_1000))

### Part b

Write a function that returns of draws from a distribution, where is the upper limit

#<Write code here>  
  
standard\_deviation\_function <- function(n = 1000, upper\_limit = 1){  
 sample <- runif(n = n,   
 min = 0,   
 max = upper\_limit)  
  
standard\_deviation <- sqrt(var(sample))  
  
return(standard\_deviation)  
   
}  
  
  
standard\_deviation\_function(n = 50, upper\_limit = 500)

[1] 163.7308

### Part c

Obtain the values against values of ranging from 1 to 10 (using any large value of of your choosing). Is there a consistent pattern? Can you propose a model that predicts with upper limit ?

results <- rep(as.numeric(NA), 10)  
#<Write code here>  
  
for (i in seq(1:10)){  
 test <- standard\_deviation\_function(n = 10000, upper\_limit = i)  
 results[i] <- test  
}  
  
results

[1] 0.2884436 0.5796213 0.8629285 1.1566274 1.4290612 1.7291592 2.0361542  
 [8] 2.3038797 2.6110149 2.8793285

results[1] - results[2]

[1] -0.2911777

results[2] - results[3]

[1] -0.2833072

results[3] - results[4]

[1] -0.293699

results[4] - results[5]

[1] -0.2724338

0.288 + 11\*.288

[1] 3.456

10\*.288

[1] 2.88

11\*.288

[1] 3.168

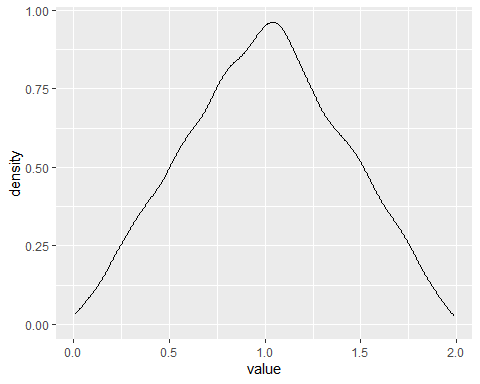
The results sample standard deviation seems to increase by approximately 0.288 as the upper limit increases by 1. I would propose that the standard deviation could be predicted by multiplying the upper limit by 0.288.

## Problem #3 - Random Variables II

### Part a

Generate two vectors and , each of which represents 10,000 draws from the distribution. Obtain the pairwise sum of these vectors () and plot the density histogram of .

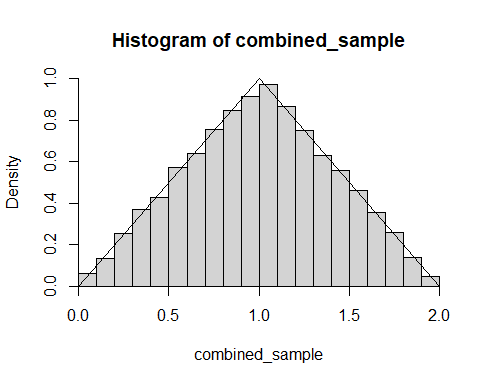
#<Write code here>  
  
sample\_1 <- runif(n = 10000,   
 min = 0,   
 max = 1)  
   
sample\_2 <- runif(n = 10000,   
 min = 0,   
 max = 1)  
combined\_sample <- sample\_1+sample\_2  
  
data <- as\_tibble(combined\_sample)  
  
density\_plot <-   
 ggplot2::ggplot(data)+  
 ggplot2::geom\_density(aes(x= value))  
  
density\_plot



### Part b

Guess what the probability distribution function (p.d.f.) of might be (i.e. describe it mathematically with a pairwise function), and illustrate it with a line over the histogram.

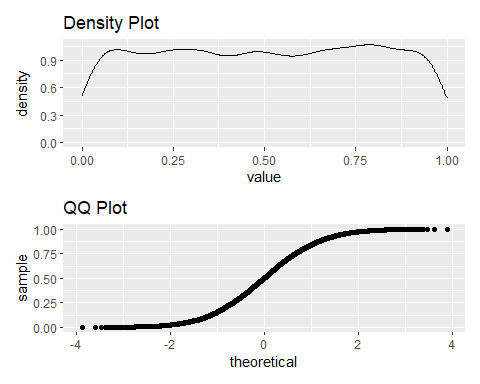
hist(combined\_sample, probability = TRUE, xaxs = "i", yaxs = "i", ylim = c(0, 1))  
lines(c(-10, 0, 1, 2, 10), c(0, 0, 1, 0, 0))

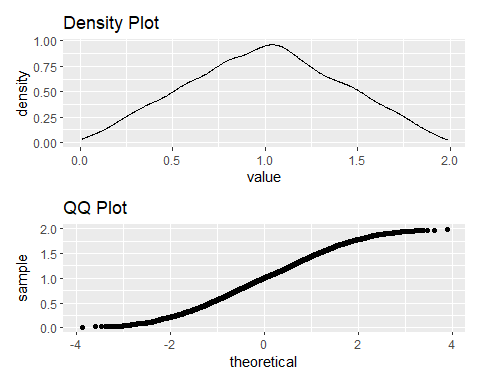


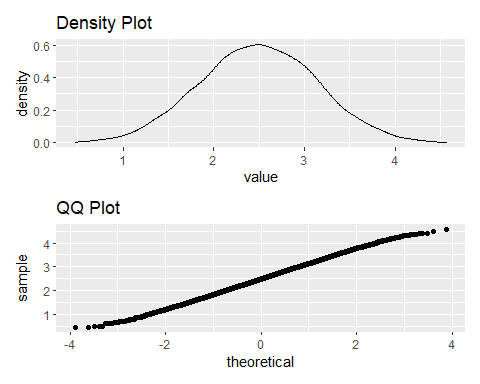
### Part c

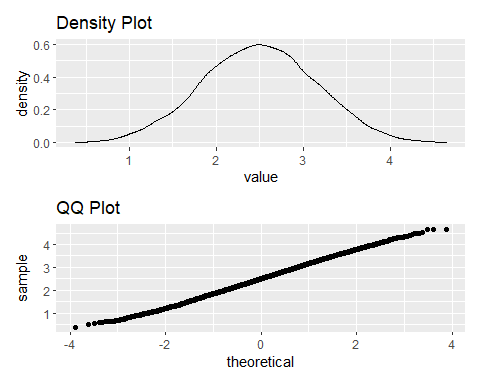
Similarly, obtain and . Plot on a single 4x2 figure the respective histograms and qqnorm plots of . What do you observe?

#<Write code here>  
  
  
combined\_sample\_5 <- runif(10000, 0, 1)  
for (i in 2:5) {  
 combined\_sample\_5 <- combined\_sample\_5 + runif(10000, 0, 1)  
}  
  
combined\_sample\_10 <- runif(10000, 0, 1)  
for (i in 2:5) {  
 combined\_sample\_10 <- combined\_sample\_10 + runif(10000, 0, 1)  
}  
  
tibble\_1 <- as\_tibble(sample\_1)  
  
tibble\_2 <- as\_tibble(combined\_sample)  
  
tibble\_5 <- as\_tibble(combined\_sample\_5)  
  
tibble\_10 <- as\_tibble(combined\_sample\_10)  
tables <- list(tibble\_1, tibble\_2, tibble\_5, tibble\_10)  
  
for(i in seq\_along(tables)){  
 # i = 1  
 density\_plot <- ggplot2::ggplot(tables[[i]])+  
 ggplot2::geom\_density(aes(x= value))+  
 labs(title ="Density Plot",   
   
 ylab = "Density")  
   
 qqplot <- ggplot2::ggplot(tables[[i]])+  
 ggplot2::geom\_qq(aes(sample= value)) +  
 labs(title = "QQ Plot ")  
   
 print( density\_plot / qqplot)  
  
  
}









As more observations are taken from the normal distribution, the data becomes more normal (the QQ plot approaches a diagonal line with a positive slope). This is an example of the Central Limit Theorem.

## Problem #4 - De Méré’s Wager I

[Antoine Gombaud](https://en.wikipedia.org/wiki/Antoine_Gombaud), a.k.a. the Chevalier de Méré (1607-1684) was a “gentleman gambler” (and amateur mathematician) in France. One game de Méré enjoyed playing (though it sounds awfully boring to me) would be to bet with even odds (i.e. lose a franc if you lose, win a franc if you win) on getting at least one six on four rolls of a fair die. He reasoned that the chance of getting a six in one roll of a die is , so the chance of getting one six in four would be .

### Part a

Write a function DeMereA(n) that simulates this game, i.e. takes a certain number of rolls of a die and returns a TRUE if there is at least one six and a FALSE if there isn’t.

#<Write code here>  
  
DeMere\_A <- function(number\_rolls = 10){  
 sample <- as.vector(c(1L:6L))  
   
 result <- sample(x = sample, size = number\_rolls,   
 replace = T)  
   
 if( 6 %in% result){  
 return(TRUE)  
 } else {  
 return(FALSE)  
 }  
 }

### Part b

Simulate 1000 of these games and compute the number of times he wins.

#<Write code here>  
  
  
DeMere\_B <- function(number\_rolls = 10){  
 sample <- as.vector(c(1L:6L))  
   
 result <- sample(x = sample, size = number\_rolls,   
 replace = T)  
   
 wins <- result[result==6]  
   
 return(length(wins))  
   
}  
  
DeMere\_B(number\_rolls = 1000)

[1] 187

### Part c

De Méré’s reasoning was faulty, but he still won a lot of money playing this (frankly, kind of stupid) game. What is the actual probability of getting at least one six in four rolls? (Use math or simulate in R)

#<Write code here>  
  
1-(5/6)^4

[1] 0.5177469

## Problem #5 - De Méré’s Wager II

Despite the faulty reasoning, de Méré actually won a lot of money playing this game but became bored of it. So, to make things mildly more interesting, he devised a second game. In this one, he would wager - again with even odds - on getting at least one double six on 24 rolls of a pair of dice. He reasoned correctly that the chance of getting a double six in rolling a pair of dice is . So he assumed that in 24 rolls of a pair of dice, the chance of getting one double six would be .

### Part a

Write an analogous function DeMereB(n) that simulates the 24 roll game with double dice.

#<Write code here>  
  
DeMere\_24 <- function(number\_rolls = 24){  
 sample <- as.vector(c(1L:6L))  
   
 result\_1 <- as\_tibble\_col(sample(x = sample,   
 size = number\_rolls,   
 replace = T),   
 column\_name = "die\_1")  
   
 result\_2 <- as\_tibble\_col(sample(x = sample,  
 size = number\_rolls,   
 replace = T),   
 column\_name = "die\_2")  
   
   
   
 wins\_1 <- bind\_cols(result\_1, result\_2) %>%   
 filter(die\_1 == 6 & die\_2 ==6)  
   
   
   
 return(nrow(wins\_1))  
   
}  
  
DeMere\_24(number\_rolls = 24)

[1] 0

### Part b

Simulate 1000 of these games and compute the number of victories.

#<Write code here>  
DeMere\_24\_results <- replicate(1000, DeMere\_24(24))  
  
  
sum(DeMere\_24\_results)

[1] 664

### Part c

Again, de Méré’s reasoning was faulty. Compute the actual probability of getting at least one double six in twenty four rolls. (Use math or simulate in R)

#<Write code here>  
1-(35/36)^24

[1] 0.4914039

Based on more empirical data (i.e. losing a lot of money), he knew something was not quite right in the second game of dice. So he challenged his famous friend [Blaise Pascal](https://en.wikipedia.org/wiki/Blaise_Pascal) to help him find an explanation. In a series of letters between Pascal and [Pierre de Fermat](https://en.wikipedia.org/wiki/Pierre_de_Fermat), not only did de Méré learn not to make naive calculations, but the foundation was laid for the theory of probability.

## Problem #6 - Card Playing

A standard playing deck of 52 cards consists of 13 unique valued cards (2 to 10, and four face cards: the Ace, Jack, Queen, and King) each in 4 different suits (hearts, diamonds, clubs, and spades). You can generate a complete deck of cards in R using something like the following code, although I recommend you simplify this deck to just what’s necessary to simulate each question.

CardNames <- c(2:10, "Jack", "Queen", "King", "Ace")  
Deck <- c(paste(CardNames, "of Diamonds"),  
 paste(CardNames, "of Hearts"),  
 paste(CardNames, "of Clubs"),  
 paste(CardNames, "of Spades"))  
  
rm(CardNames)  
  
print(c(head(Deck), tail(Deck)))

[1] "2 of Diamonds" "3 of Diamonds" "4 of Diamonds" "5 of Diamonds"   
 [5] "6 of Diamonds" "7 of Diamonds" "9 of Spades" "10 of Spades"   
 [9] "Jack of Spades" "Queen of Spades" "King of Spades" "Ace of Spades"

And then we can “draw” five cards simply with: sample(Deck, 5, replace = TRUE).

### Part a

How many possible hands of five unique cards can be drawn from a deck of cards? (Note: cards are drawn without replacement)

factorial(52)/ factorial(52-5)/factorial(5)

[1] 2598960

### Part b

In poker, a flush is defined as five cards of the same suit, regardless of the value of the card. Write a function called IsFlush() that draws five cards from a deck, determines whether or not it is a flush, and returns the appropriate logical value.

#<Write code here>  
is\_flush <- function(number\_cards = 5){  
   
 hand <- as\_tibble\_col(sample(Deck, number\_cards, replace = FALSE), column\_name = "hand")  
   
 suites <- hand %>%   
 mutate(suite = sub(".\*of ","", hand ))  
   
 unique\_suites <- unique(suites$suite)  
   
 if(length(unique\_suites) ==1){  
 return(TRUE)  
 }else{  
 return(FALSE)  
 }  
   
}  
  
  
is\_flush(number\_cards = 5)

[1] FALSE

### Part c

Simulate this process 10,000 times and count how many times you draw a flush. What is the approximate probability of a flush based on this experiment? (Note: this can be approached with a loop or apply() on a matrix of decks)

#<Write code here>  
  
flush\_simulation <- replicate(10000, is\_flush(number\_cards = 5))  
  
  
sum(flush\_simulation[flush\_simulation==TRUE])

[1] 23

Based on this experiment, the chance of drawing a flush is 0.0022%. My function showed a total of 22 flushes in 10,000 simulations.

### Part d

A straight is a sequence of five cards in numerical order (note: for simplicity we define Ace = 13 only. So Ace-2-3-4-5 is not considered a straight). Repeate exercises (b) and (c) for the straight (i.e. create a function IsStraight() and repeat the experiment 10,000 times)

#<Write code here>  
  
is\_straight <- function(number\_cards = 5){  
   
 hand <- as\_tibble\_col(sample(Deck, number\_cards, replace = FALSE), column\_name = "hand")  
   
 numbers <- hand %>%   
 mutate(card\_value = sub(" of.\*","", hand )) %>%   
 mutate( card\_value\_numeric =   
 case\_when(card\_value == "Jack"~ "11",   
 card\_value== "Queen"~ "12",  
 card\_value== "King"~ "13",   
 card\_value== "Ace"~ "14",  
 TRUE ~ card\_value)  
 ) %>%   
 mutate(card\_value\_numeric = as.numeric(card\_value\_numeric)) %>%   
 arrange(card\_value\_numeric) %>%   
 mutate(value\_difference = card\_value\_numeric -lag(card\_value\_numeric))  
   
   
  
   
 if(sum(numbers$value\_difference, na.rm = T ) ==4){  
 return(TRUE)  
 }else{  
 return(FALSE)  
 }  
   
}  
  
straight\_simulation <- replicate(10000, is\_straight(number\_cards = 5))  
  
  
sum(straight\_simulation[straight\_simulation==TRUE])

[1] 245

### Part e

Based on these results, which is the more likely hand to be dealt in a game of poker?

My function returned 252 straights in 10,000 draws (an approximate probability of 0.0252%). It’s much more likely that you will get a straight than a flush.

## Problem #7 - Airplane Functioning

You are planning on taking a flight on Epsilon Airlines across the Pacific Ocean. To have a successful flight, 100 different components on the plane must ALL function correctly. Each of these components has a probability 0.001 of failing.

### Part a

Use the rbinom() function to simulate a flight on Epsilon airline by creating a vector of 100 elements which succeed or fail with the appropriate probability.

#<Write code here>  
airplane\_parts <- rbinom(100,1, 0.001)

### Part b

Use the rbinom() function in a slightly different way to produce a vector of length 1,000,000 in which each element is the number of components that fail in each flight. Use this vector to produce an estimate of the probability of failure for the plane.

#<Write code here>  
part\_failures <- rbinom(1000000, 100, 0.001)  
  
failure\_odds <- mean(part\_failures > 0)

### Part c

Based on this result, would you take a flight on Epsilon airlines? Why or why not?

No, definitely not. The probability of something going wrong with the plane is 0.09566. That’s well above my level of risk tolerance.

### Part d

Using dnorm(), calculate the exact probability that the flight will not be successful. What important assumption are you making in this calculation (and the previous simulations)?

#<Write code here>  
sum(dbinom(1:100, 100, 0.001))

[1] 0.09520785

The major assumption that we have made in this problem is that the failure of one part will not cause the failure of other parts. We are assuming that all parts in this airplane are independent of each other.