---
title: "Chapter 2-Probability"
output:
  html_document

Simulation

<https://media.csuchico.edu/media/Math+350+Section+2_2+Part+1/1_5oeaiwz6>

During this course we will be solving problems numerically and also via simulation. This RMarkdown file is where we will do the simulation problems. Any problem in this course that we will encounter can be solved numerically. Sometimes this may require calculus or other fancy mathematical methods to solve a problem. An alternative way to solve almost any probability problem is via simulation. Simulation ``simulates'' a mathematical problem by using repeated sampling from a population or sample space.

Simulations with `sample`

The R command `sample` can be used to simulate sampling from a bunch of outcomes. Think of it as putting names in a hat and individually drawing the names out of the hat. When you use the command `sample` you need to give a vector to sample from and also the sample size. The default is to sample **without replacement** with each item having equal probability of being selected.

What does sampling **without replacement** mean?

The defaults can be changed. For instance, if you want to sample **with replacement** you would just add replace=TRUE to the command.

```{r}
##Examples using sample

##Take a sample of 2 from the numbers 1 through 10.

x<-1:10
(sample(x,2)) #take a sample of size 2 from vector x
 (sample(x,20,replace=TRUE)) #vector size you want to sample and true for replace, the default is false
##Sample three months from the list of months

months<-c("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sept","Oct","Nov","Dec")

sample(months,3)
```

Example 2.9

<https://media.csuchico.edu/media/Math+350+Chapter+2_2+Part+2/1_fkbskp6g>

In the United states, human blood comes in four types: O,A,B,AB. Take a sample of thirty blood types with the probabilities given in the example.

To do this we need to change two of the defaults. Obviously, if we want to sample 30 blood types we would need to sample with replacement. Additionally, we want to sample in a way that selects the blood types consistent with their probabilities. To do this we can create a vector of probabilities and enter it into our `sample` function.

```{r}
bloodtypes<-c("O","A","B","AB")
prob_bloodtypes<-c(0.45,.4,.11,.04)
sample(bloodtypes,30,prob=prob_bloodtypes,replace=TRUE) # (vector with data to sample, times to samples, set the prob attribute, and replacement is required so the option is a possibility on the next trial) #set.seed allows us to control outcomes, in class we didn't get the same sample as each other if you want to use it later you must save it to variable.
```

If we were to take a very large sample, we would expect the proportion of each blood type in the sample to mimic the probabilities of each.

```{r}
library(dplyr)

n<-10000
sample_blood<-sample(bloodtypes,n,prob=prob_bloodtypes,replace=TRUE)
#(sample_blood<-sample(bloodtypes,1000,prob=prob_bloodtypes,replace=TRUE))

#(sample_blood<-sample(bloodtypes,10000,prob=prob_bloodtypes,replace=TRUE))
table(sample_blood)%>%prop.table()#%>% take the table and make it a porportion
table(sample_blood)/n



```

Using simulation to compute probabilities

<https://media.csuchico.edu/media/Math+350+Chapter+2_2+Part+3/1_dhrp5i9n>

The goal of simulation is to compute probabities of an event. We can do this in three steps:

1. Simulate the experiment many times storing the results in a vector.
2. Test of the outcome that we are interested in is in the event and store as TRUE/FALSE.
3. Compute the proportion of TRUEs to figure out the probabity.

Example 2.10
Suppose that two six-sided dice are rolled and the numbers appearing on the dice are added. Simulate this experiment by performing 10000 rolls of each die and then adding the die together.

```{r}
dice<-1:6#sample space
n<-10000
die1<-sample(dice,n,replace=TRUE)
die2<-sample(dice,n,replace=TRUE)
results<-die1+die2
table(results)/n

number_to_add<-sample(dice,10000,replace= TRUE)
number_to_Be_added<-sample(dice,10000,replace= TRUE)

answer<-sum(number_to_add<-sample(dice,10000,replace= TRUE),nnumber_to_Be_added<-sample(dice,10000,replace= TRUE))

dice1<-sample(1:6,10000,replace=TRUE)
dice2<-sample(1:6,10000,replace=TRUE)

sim_results<-dice1 + dice2
sim_results[1:10]

```

Let *E* be the event *the sum of the dice is 6* and *F* the event *At least one of the dice is a 2*.

We can get the probability of these by first extracting each event from our simulation results.

```{r}
E<-sim_results ==6
E[1:20]

mean(E)


```

E and F now consists of TRUEs and FALSEs. If we want to change this into a numeric variable the FALSEs would correspond to 0 and the TRUEs would correspond to 1. If we use the command `mean` then R automatically turns the vector into a numeric vector of 0s and 1s and then calculates the mean. When we have 0 and 1 data, the average is equivalent to the proportion of 1s. Thus, we can compute the probability of event E or F by calculating the mean.

```{r}
f<-dice1==2 | dice2==2
f[1:20]

mean(f)

```

Example

<https://media.csuchico.edu/media/Math+350+Chapter+2_2+Part+4/1_22mugooo>

Suppose the proportion of M&Ms by color is: 14% yellow, 13% Red, 20% Orange, 12% Brown, 20% Green, and 21% Blue.

Answer the following questions (without using simulation):

a. What is the probability that a randomly selected M&M is not green?
P(Not Green)= 1.0-0.2 = 0.8

b. What is the probability that a randomly selected M&M is red, orange, or yellow?
P(Red, orange or yellow)=0.13+0.2=0.14=0.46

Repeat the above exercise using simulation.

```{r}
sample_space<-c("yellow","red","orange","brown","green","blue")
color_prob<-c(.14,.13,.2,.12,.2,.21)

sim_results <- sample(sample_space,10000,replace=TRUE, prob=color_prob)

A<- sim_results =="green"
B <- sim_results == "red" |sim_results =="orange" |sim_results=="yellow"

(prob_A = 1-mean(A))
(prob_B = mean(B))
```

```

Are the exact probabilities and the simulated probabilities close?


## Using `replicate` to repeat experiments

The function `replicate` can be used to repeat an experiment many times. Although we used `sample` in the above examples, `replicate` might be necessary for more complicated experiments. Once you create code to do something one time, you wrap that code inside of curly brackets and use the function `replicate` to repeat it.

```{r}
##Simulate rolling a dice 7 times and computing the sum of the 7 rolls and recording whether the sum is graded then 7.
dice_7<- 1:6
answer<-sum(sample(dice_7,7,replace=TRUE))

dice<-1:6
dice_7<-sample(dice,7,replace=TRUE)
results<-sum(dice_7) >7
results


```

What if we wanted to repeat the above experiment several times? We can keep clicking the *run* button and record the results each time. However, `replicate` will do this for us and much more efficiently. To use the function `replicate` we just wrap the above code in brackets and tell R how many times we want to repeat (or replicate) the experiment of tossing a die 7 times and calculating the sum.

```{r}
replicate(20,{
  dice_7<-sample(dice,7,replace=TRUE)
  results<-sum(dice_7) >7
})
```

We will almost always want to replicate things a large number of times, say 10000. We should store the output in a vector.
```{r}

```

Now, calculate the probability of rolling a die 7 times and getting a sum larger than 30.

```{r}
dice<-1:6
results_dice<-replicate(10000,{
  dice<-sample((dice), size=7,replace=TRUE)
  sum(dice)>30
})

mean(results_dice)


```

The following sequence is how you should approach writing code that uses the function `replicate`:

1. Write code that performs the experiment a single time.
2. Replicate the experiment a small number of times and check the results.
3. Replicate the experiment a large number of times and store the results.
4. Compute probability using the `mean` of the results.

You try it

If two die are rolled, what is the probability that the difference between the two numbers is less than 3?

```{r}
#Step 1: Write code that performs the experiment a single time. You may need to use the function abs which takes the absolute value of a number.

dice<-1:6 #declare a dice that is six sided

dice_uno<-sample((dice),size=7,replace=TRUE) #dice 1
dice_dos<-sample((dice),size=7,replace=TRUE) #dice 2

difference<-(dice_uno-dice_dos < 3 | dice_dos-dice_uno< 3) #find the difference between two dice if it is less than 3

difference #if you get a bunch of true it is because you haven't replicated!
#Step 2: Replicate the above experiment a small number of times, say 20

results<- replicate(20,{

```
  dice<-1:6 #declare a dice that is six sided

  dice_uno<-sample((dice),size=7,replace=TRUE) #dice 1
  dice_dos<-sample((dice),size=7,replace=TRUE) #dice 2

  difference<-(dice_uno-dice_dos < 3 | dice_dos-dice_uno< 3) #find the difference between two dice if it is less than
3
})

mean(results)

##Step 3: Up the replicates and store them in a vector
results<- replicate(10000,{
  dice<-1:6 #declare a dice that is six sided

  dice_uno<-sample((dice),size=7,replace=TRUE) #dice 1
  dice_dos<-sample((dice),size=7,replace=TRUE) #dice 2

  difference<-(dice_uno-dice_dos < 3 | dice_dos-dice_uno< 3) #find the difference between two dice if it is less than
3
})

##Take mean of results to determine probability
mean(results)

```

#### You try it

After reviewing Example 2.13 try the following problem:

A fair coin is repeatedly tossed ten times. Estimate the probability that you observe Heads on the last three tosses.

```{r}
coin<-sample(1:0,1) #heads is 1 and tails is 0
results<-replicate(10,{
  coin<-sample(0:1,1)
})

results

mean(results)

### Alternative WAy toss<-c("heads","tails")
toss<-c(0,1)
results_toss<-sample(toss,10,replace=TRUE)
results_toss[8:10] ==1

sum(results_toss[8:10])==3

#step 2
results<-replicate(10000,{
  toss<-c(0,1)
  results_toss<-sample (toss,10,replace=TRUE)
  results_toss[8:10] ==1
  sum(results_toss[8:10])==3
})

mean(results)

### Alternative Way
#results<-replicate(10000,{
#toss<-c(0,1)
#results_toss<-sample(toss,10,replace=TRUE)
#sum(results_toss[8:10])
#})
```

Now compute the probability by hand that the last three coin tosses results in **heads**.

End of Chapter Problems

Use simulation to solve the following problems:

1. Blood types O, A, B, and AB have the following distribution:
```{r table1, echo=FALSE,message=FALSE,warning=FALSE}
tabl <- "
|-----|-----|-----|-----|-----|
| Type| O   | A   | B   | AB  |
|-----|-----|-----|-----|-----|
|Prob |0.45 | 0.40|0.11 | 0.04|
|-----|-----|-----|-----|-----|
"
```

```
cat(tabl)
```
What is the probability that two randomly selected people have the same blood type?

```{r}
blood_types <-c("AB","A","B","O") #sample space
prob_bloodtypes<-c(0.04,0.40,0.11,0.45) #probability vector
person1<-sample(blood_types,10000,replace = TRUE, prob= prob_bloodtypes)
person2<-sample(blood_types,10000,replace = TRUE, prob= prob_bloodtypes)
#sample 2 people from the sample space.
results <- person1 == person2

mean(results)

```

2. In a room of 200 people (including you), estimate the probability that at least one other person will be born on the same day as you.

```{r}

results<-replicate(10000,{
days<-1:365
myBday<-23
randoGuy<- sample(days,199,replace=TRUE)
birthdayTwins<-sum(myBday == randoGuy[1:199])>0

})

mean(results)

```

3. If 100 balls are randomly placed into 20 urns, estimate the probability that at least one of the urns is empty.

```{r}
ballfinal <- replicate(10000, {

  balls<-1:20
  urns<-sample(balls,100, replace=TRUE)
  results<-length(unique(urns))< 20
})

mean(ballfinal)
```
4. A standard deck of cards has 52 cards, four each of 2,3,4,5,6,7,8,9,10, J,Q,K,A. In blackjack, a player gets two cards and adds their values. Cards count as their usual numbers, except Aces are 11 (or 1), while K, Q, J are all 10. Use R to simulate dealing two cards, and compute these probabilities

A **blackjack** means getting an Ace and a value ten card. What is the probability of getting a blackjack?

```{r}

deck<-c(rep(2:10,4),rep(10,12),rep(11,4))

length(deck) #confirming we have a full 52 card deck

results<-replicate(10000,{
  p1<-sum(sample(deck,1),sample(deck,1))

  blackjack <-21
  winners<-p1 ==blackjack })

  mean(results)

```

What is the probability of getting 19? Assume that an Ace counts as 11.

```{r}
deck<-c(rep(2:10,4),rep(10,12),rep(11,4))

results<-replicate(10000,{
  p1<-sum(sample(deck,1),sample(deck,1))

  blackjack <-19
  winners<-p1 ==blackjack })

  mean(results)

```

5. Suppose the proportion of M&Ms by color is: 14% yellow, 13% Red, 20% Orange, 12% Brown, 20% Green, and 21% Blue. Also, suppose that you buy a bag of M&Ms with 30 pieces in it. Estimate the probability of obtaining at least 9 Blue

M&Ms and at least 6 Orange M&Ms in the bag.

```{r}
snackBag<- replicate(10000,{
  colors<-c("yellow","red","orange","brown","green","blue")
probability<-c(.14,.13,.2,.12,.2,.21)

trials <-sample(x=colors,30,prob=probability,replace=TRUE)

results<-sum(trials=="blue")>8 & sum(trials== "orange")>5})

mean(snackBag)

```

6. Deathrolling in World of Warcraft works as follows: Player 1 tosses a 1000 sided die. Say they get x_{1}. Then player 2 tosses a die with x_{1} sides on it. Say they get x_{2}. Player 1 tosses a die with x_{2} sides on it. This pattern continues until a player rolls a 1. The player who loses is the player who rolls a 1. Estimate via simulation the probability a 1 will be rolled on the 4th roll in deathroll.

```{r}
deathrool<-replicate(100000,{
  y=TRUE
  rollUno<-sample(1:1000,1,replace=TRUE)
  rollDos<-ifelse(rollUno==1, FALSE,sample(1:rollUno,1,replace=TRUE))
  rollTres<-ifelse(rollDos==1,FALSE,sample(1:rollDos,1,replace=TRUE))
  rollQuatro<-ifelse(rollTres==1,FALSE,sample(1:rollTres,1,replace=TRUE))
  y<-rollQuatro==1
  y

})

table(deathrool)
mean(deathrool)

```
Teacher's Advice

```
x<-c(1,2,3)
y <- ifelse( if, true, false)

y<-(sum(x)==6,1,0)
```

An if else statement, if the sum of x is equal to 6 assign 1 to y otherwise it is false and assign it to a 0

```
x<-c(1,2,3)
y<-(sum(x) ==3, 2, ifelse(sum(x)==6,1,0))
```