Homework 1

1. History of Probability

It is said that de Mere had been betting that, in four rolls of a die, at least one six would turn up. He was winning consistently and, to get more people to play, he changed the game to bet that, in 24 rolls of two dice, a pair of sixes would turn up. It is claimed that de Mere lost with 24 and felt that 25 rolls were necessary to make the game favorable.

In [1]:

```
#1. Write function to sample from die
sample_die <- function()
{
    # randomly choose a number between 1 and 6
    sample(1:6,1, replace=T)
}

#2. Write a loop function to tally sucesses vs failures (define
sucesses?)
suc_fail <- function(n)
{
    success <- 0
    failure <- 0

    for (i in 1:n){
        a <- sample_die()
        b <- sample_die()
        if (a == 6 & b == 6){
            success <- success + 1
        } else {
            failure <- failure + 1
        }
    }
    result <- list(success, failure)
    return (result)
}

#3. Run 10000 trials (24 throws 10000 times - 25 throws 10000 ti
```

```
mes) keeping track of successes vs faiures (from function 2)

trials <- function(times, throws)
{
    successes <- 0
    failures <- 0

    for (k in 1:times){
        trial_res <- suc_fail(throws)

        if (trial_res[[1]] >= 1){
            successes <- successes + 1
        } else{
            failures <- failures + 1
        }
    }

    final_result <- list(successes, failures)
    return (final_result)
}

a <- trials(10000,24)
b <- trials(10000,25)

cat("Number of successes for 24 throws: ", a[[1]], " ")
cat("Number of failures for 24 throws: ", a[[2]], " ")
cat("Number of successes for 25 throws: ", b[[1]], " ")
cat("Number of failures for 25 throws: ", b[[2]], " \n")

#4. Calculate probabilities for 24 rolls and 25 rolls

prob_t4 = 1 - ((35/36)**24)
cat("Probability for 24 rolls: ", prob_t4, " ")
prob_t5 = 1 - ((35/36)**25)
cat("Probability for 25 rolls: ", prob_t5, " ")


#5. Answer if de Mere was right and that we will win with 25 thr
ows, give a mathematical explanation.
# de Mere was right that we have a better chance of winning the
game with 25 throws. The chance of winning
# is slightly over 50% when there are 25 throws. When there are
only 24 throws, the chance of winning
# is slightly under 50%.
# Mathematically, it makes sense that there's a higher chance of
```

```
success when there's 25 throws.
# The probability formula creates larger probabilities for more
throws.
# The number of throws is equal to the exponent that is applied
to the fraction of failures over possible outcomes.
# After this term is evaluated, it is subtracted from 1 to obtai
n the probability.
# Therefore, larger exponents makes the term smaller. As a resul
t, a smaller value is subtracted from 1, and the
# probability is larger.
```

```
Number of successes for 24 throws:  4871  Number of
failures for 24 throws:  5129  Number of successes f
or 25 throws:  5118  Number of failures for 25 throw
s:  4882
Probability for 24 rolls:  0.4914039  Probability fo
r 25 rolls:  0.5055315
```

2. Addition rule

Find the probabilities using the table

1. Type O or AB.
2. Type A or AB.
3. Type AB or Rh negative.
4. Type O and Negative.
5. Type AB

Type	O	A	B	AB	Total
Positive	163	662	1513	1603	3941
Negative	224	933	2400	2337	5894
Total	387	1595	3913	3940	9835

In [2]:

```
#1 Type O or AB
o_ab = (387+3940)/(9835)
cat("Probability of Type O or AB: ", o_ab, " \n")

#2 Type A or AB
a_ab = (1595+3940)/(9835)
cat("Probability of Type A or AB: ", a_ab, " \n")

#3 Type AB or Rh negative
ab_rhn = (224+933+2400+2337+1603)/(9835)
cat("Probability of Type AB or Rh negative: ", ab_rhn, " \n")

#4 Type O or negative
o_neg = (163+224+933+2400+2337)/(9835)
cat("Probability of Type O or Negative: ", o_neg, " \n")

#5 Type AB
ab = (3940)/(9835)
cat("Probability of Type AB: ", ab, " \n")
```

```
Probability of Type O or AB:  0.4399593
Probability of Type A or AB:  0.562786
Probability of Type AB or Rh negative:  0.7622776
Probability of Type O or Negative:  0.6158617
Probability of Type AB:  0.4006101
```

3. Multiplication Rule

Answer questions below using this exercise: If I roll 5 dice, what is the chance of getting all sixes? What is the chance of getting no sixes?

Write a simulation in R to obtain the probabilities for each of these two exercises running 10000 trials

Hint.

Remember that the change to get all 6 in 5 rolls is $(1/6)^5$

and to get no sixes is $(5/6)^5$

In [3]:

```
#All sixes
#1. Write function to sample from die
sample_die <- function()
{
    # randomly choose a number between 1 and 6
    sample(1:6,1, replace=T)
}

#2. Write function to determine if in each 5 rolls we get all si
xes (6*5)

allsixes <- function(n) {
    suc = 0

    for (k in 1:5){
        one_d = sample_die()

        if (one_d == 6){
            suc = suc + 1
        }
    }
    if (suc == 5){
        return("success")
    }
    else{
        return("failure")
    }
}

#3. Run 10000 trials - Hint check sapply function

vec <- 1:10000

a <- sapply(vec, allsixes)

#print(a)

succ = 0
for (p in 1:length(a)){
    if (a[[p]] == 'success'){
        succ = succ + 1
    }
}
```

```
cat("Number of trials that had all sixes: ", succ, " \n")

#4. Calculate probabilities
allsixes = ((1/6)**5)
cat("Probability of getting all sixes: ", allsixes, " \n")

#No sixes
#Repeat 1
sample_die <- function()
{
    # randomly choose a number between 1 and 6
    sample(1:6,1, replace=T)
}

#Repeat 2
nosixes <- function(n) {
    suc = 0

    for (k in 1:5){
        one_d = sample_die()

        if (one_d != 6){
            suc = suc + 1
        }
    }
    if (suc == 5){
        return("success")
    }
    else{
        return("failure")
    }
}

#Repeat 3
vec2 <- 1:10000

b <- sapply(vec2, nosixes)

#print(b)

succ2 = 0
for (k in 1:length(b)){
    if (b[[k]] == 'success'){
        succ2 = succ2 + 1
```

```
    }
}

cat("Number of trials that had no sixes: ", succ2, " \n")

#Repeat 4
nosixes = ((5/6)**5)
cat("Probability of getting no sixes: ", nosixes, " \n")
```

```
Number of trials that had all sixes:  0
Probability of getting all sixes:  0.0001286008
Number of trials that had no sixes:  4057
Probability of getting no sixes:  0.4018776
```

4. Conditional Probability

Consider a family that has three children. We are interested in the children's genders. Our sample space is S={(G,G,G),(G,G,B),(G,B,G),(G,B,B),(B,G,G),(B,G,B),(B,B,G),(B,B,B)}. Also assume that all eight possible outcomes are equally likely.

1. What is the probability that the three children are girls given that the first child is a girl?
2. What is the probability that At least two children are boys given that the first child is a boy?

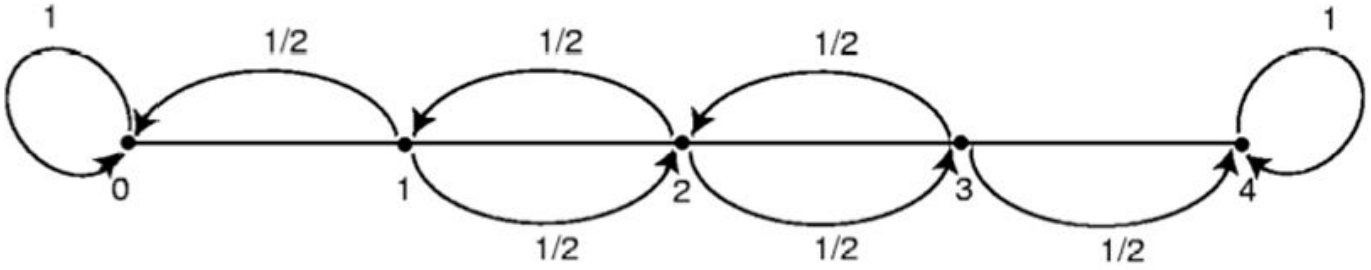
In [4]:

```
#1. Three children are girls given that the first child is a gir
l
prob = (1/8)/(0.5)
cat("Probability of having three girls given the first child is
a girl: ", prob, " \n")

#2 Probability that At least two children are boys given that th
e first child is a boy
prob2 = (3/8)/0.5
cat("Probability of having at least two boys given the first chi
ld is a boy: ", prob2, " \n")
```

```
Probability of having three girls given the first ch
ild is a girl:  0.25
Probability of having at least two boys given the fi
rst child is a boy:  0.75
```

5. Markov Chains

As explained in class, create a function that simulates the following random walk based on the conditional probabilities from the transition matrix:



Once the function is created, simulate 500 random walks of 10 steps each (you choose the initial state - can be a constant or a user defined variable).

Example of ONE random walk 2,3,2,3,4,4,4,4,4,4 (remember that 0 and 4 are absorbing states).

Plot the 500 simulations.

Capture each one of the 500 walks last state (at step 10) and create a table of frequencies for each one of the states. it should looks something like

State	Count	Frequency
0	100	0.2
1	100	0.2
2	100	0.2
3	100	0.2
4	100	0.2
total	500	1.0

In [5]:

```
##This is an example of the code, but you can create the functio
ns and the plots anyway you want (even in python)
```

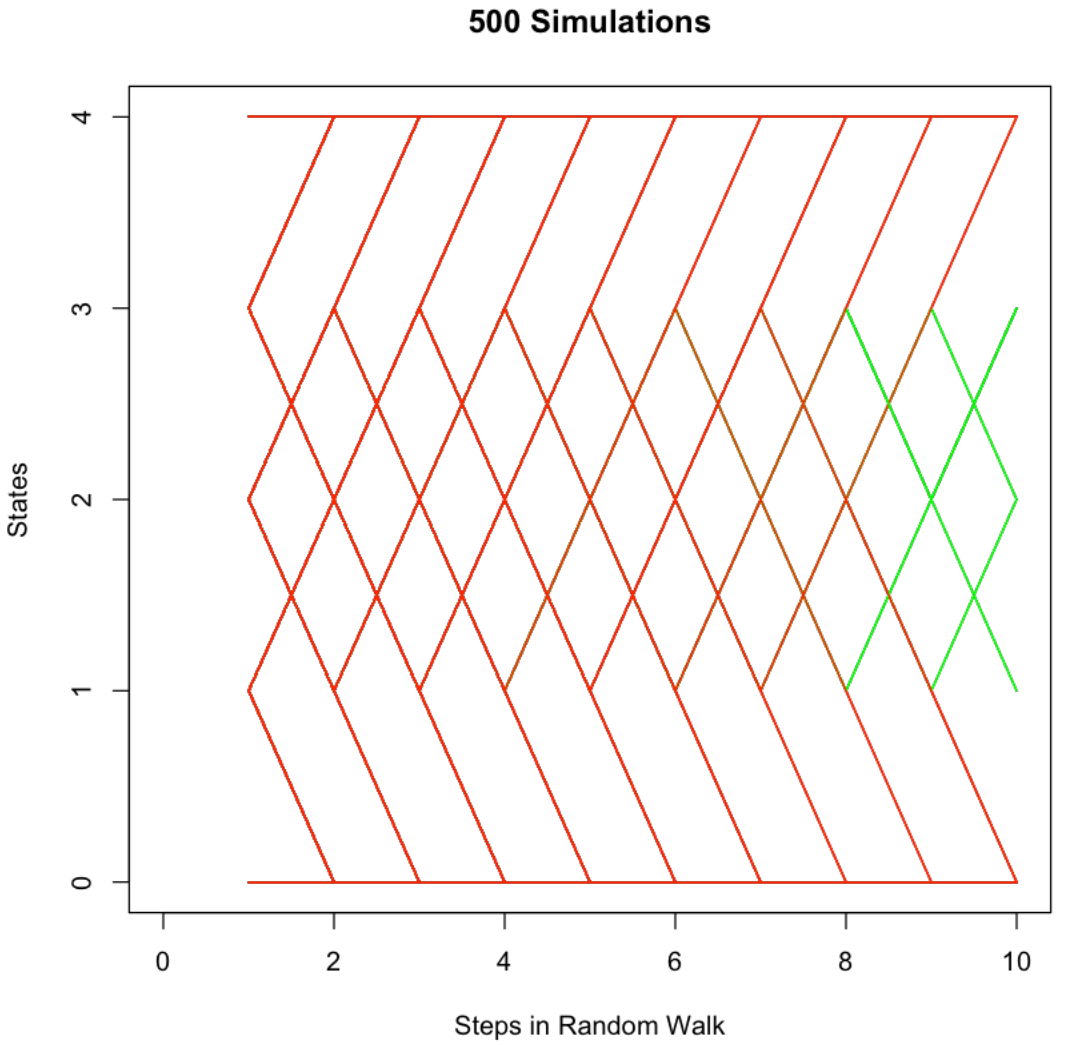
```
library(markovchain)

#1. create function that produces a random walk of size N

####Random Walk function
#N = number of steps in the walk
#Ini_prob = initial state
Sim_Markov=function(N,Ini_prob) {
  #write function that generates one random walk sampling from t
he initial state....

posStates <- c("0", "1", "2", "3", "4")
byRow <- TRUE

## transitional matrix
posMatrix <- matrix(data = c(1, 0, 0, 0, 0,
                             0.5, 0, 0.5, 0, 0,
                             0, 0.5, 0, 0.5, 0,
                             0, 0, 0.5, 0, 0.5,
                             0, 0, 0, 0, 1), byrow = byRow, nrow
= 5,
                    dimnames = list(posStates, posStates))

##make the matrix a markovchain class
mcPos <- new("markovchain", states = posStates, byrow = byRow,
             transitionMatrix = posMatrix, name = "State")

##define the initial state
initialState <- sample(c(0, 1:4),1,replace=TRUE, prob = Ini_prob
)
result <- c(as.character(initialState))

     for (k in 1:(N-1)){
        pos_matrix_row = posMatrix[(initialState+1),]

        initialState <- sample(c(0, 1:4),1, replace=TRUE, prob =
pos_matrix_row)

        result <- c(result, ",", as.character(initialState))
        }

    return(result)
}
```

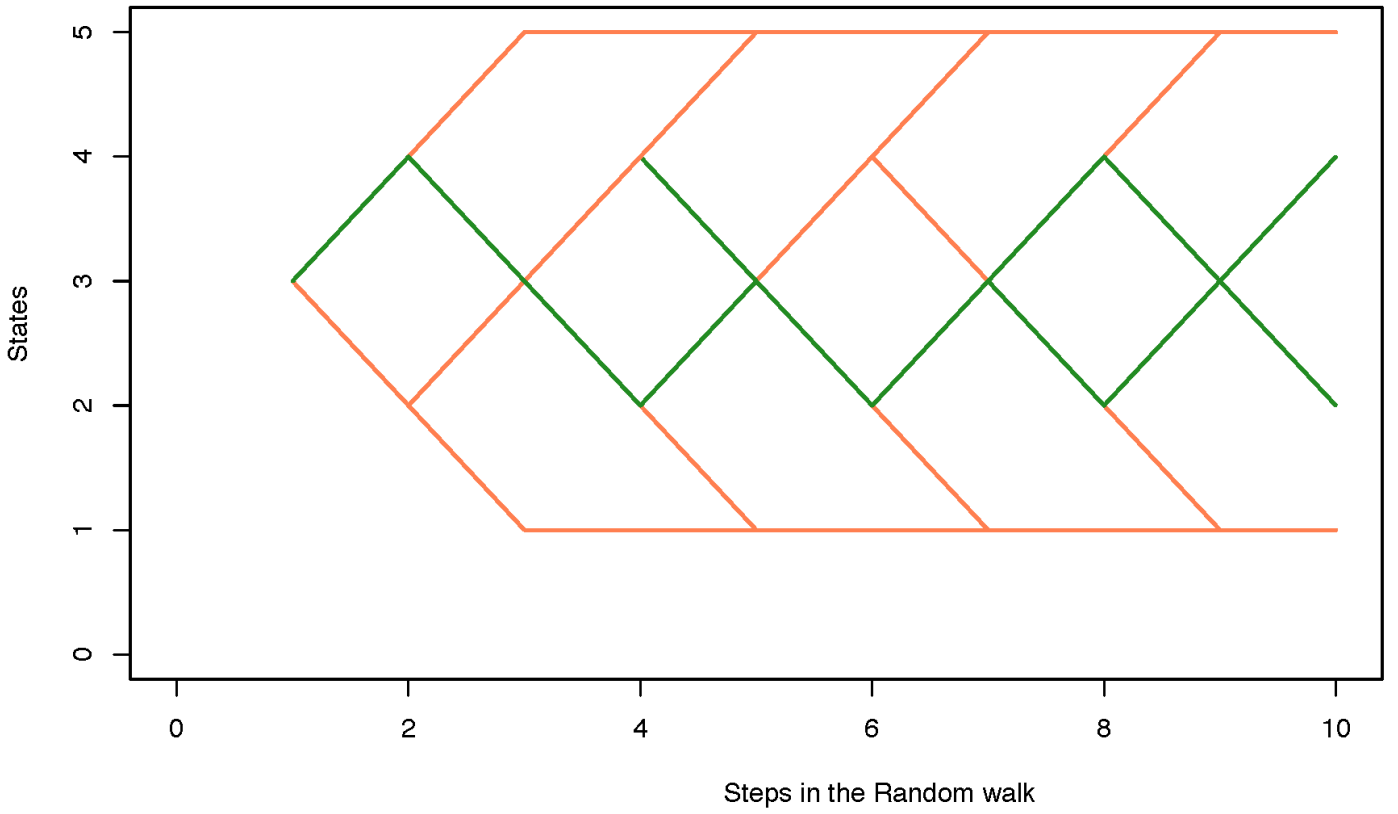
```r
#2. Define the initial state Ini_prob - different examples
Ini_prob1=c(0.0005,0.4,0.199,0.4,0.0005)
Ini_prob2=c(0.2,0.2,0.2,0.2,0.2)
Ini_prob3=c(0,0,1,0,0)

#3. Define the transition probabilities matrix
P_mat=matrix(0,5,5)
P_mat[1,]=c(1,0,0,0,0)
P_mat[2,]=c(0.5, 0, 0.5, 0, 0)
P_mat[3,]=c(0, 0.5, 0, 0.5, 0)
P_mat[4,]=c(0, 0, 0.5, 0, 0.5)
P_mat[5,]=c(0, 0, 0, 0, 1)

#Number of steps
Nst<-10

X<-Sim_Markov(Nst,Ini_prob1)
Y<-Sim_Markov(Nst,Ini_prob2)
Z<-Sim_Markov(Nst,Ini_prob3)

cat("Simulation1: ", as.character(X), " \n")
cat("Simulation2: ", as.character(Y), " \n")
cat("Simulation3: ", as.character(Z), " ")

#e.g. 3,2,1,1,1,1,1,1,1,1
```

```
Package:  markovchain
Version:  0.8.2
Date:     2020-01-10
BugReport: http://github.com/spedygiorgio/markovchai
n/issues



Simulation1:  3 , 4 , 4 , 4 , 4 , 4 , 4 , 4 , 4 , 4
Simulation2:  2 , 1 , 2 , 1 , 0 , 0 , 0 , 0 , 0 , 0
Simulation3:  2 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
```

In [6]:

```
##plot the 500 simulations
options(warn = -1)
array_sim <- c(1,2,3,4,5,6,7,8,9,10)
array <- c()
Ini_prob <- c(0,0,1,0,0)

for (k in 1:501){
    a <- runif(1)
    Ini_prob=c(0.1,sample(1:100,1, replace=T)/100,sample(1:100,1
, replace=T)/100,
               sample(1:100,1, replace=T)/100,sample(1:100,1, re
place=T)/100)
    Nst = 10
    array <- as.numeric(Sim_Markov(Nst, Ini_prob))
    array_sim <- c(array_sim, array[!is.na(array)])
}

matrix_sim <- matrix(array_sim, nrow= 501, ncol = 10, byrow = TR
UE)

matrix_sim.df <- as.data.frame(matrix_sim)

plot(matrix_sim[1,], matrix_sim[2,], main = "500 Simulations", x
lab = "Steps in Random Walk", ylab="States",
    type = "l", xlim=c(0, 10), ylim=c(0, 4), col = "red")


for (p in 3:501) {
    if (('0' %in% matrix_sim[p,]) | ('4' %in% matrix_sim[p,])){
        lines(matrix_sim[1,],matrix_sim[p,], col="red", type = "
l")
    }
    else{
        lines(matrix_sim[1,],matrix_sim[p,], col="green", type =
"l")
    }
}

##an option is to look something like this, where I am plotting
each random walk and change the color to red
##if the state is 0 or 5, green for the others.
```


500 Simulations
States
0
1
2
3
4
0
2
4
6
8
10
Steps in Random Walk


States
0
1
2
3
4
5
0
2
4
6
8
10
Steps in the Random walk

In [7]:

```
###Create table of frequencies
library("data.table")
data <- matrix_sim.df[,10]
table <- data[-1]
count0 <- length(which(table == 0))
count1 <- length(which(table == 1))
count2 <- length(which(table == 2))
count3 <- length(which(table == 3))
count4 <- length(which(table == 4))
total <- length(table)

frequency0 <- count0/(length(table))
frequency1 <- count1/(length(table))
frequency2 <- count2/(length(table))
frequency3 <- count3/(length(table))
frequency4 <- count4/(length(table))
totalf <- 1

table <- data.table("States" = c("0","1","2","3","4", "total"),
"Count" = c(count0, count1, count2, count3, count4, total),
                    "Frequency" = c(frequency0, frequency1, freq
uency2, frequency3, frequency4, totalf))
table
```

A data.table: 6 × 3

States	Count	Frequency
<chr>	<int>	<dbl>
0	192	0.384
1	3	0.006
2	6	0.012
3	8	0.016
4	291	0.582
total	500	1.000