Stochastic Modeling Projects

Kevin Mei

December 3, 2017

Packages:

```
suppressMessages(library(markovchain))
```

```
## Warning: package 'markovchain' was built under R version 3.4.2
```

```
suppressMessages(library(matrixcalc))
```

```
## Warning: package 'matrixcalc' was built under R version 3.4.1
```

Q1)

Procedure:

1.Creating two vectors X and Y that generate randomly all possible numbers within the interval from -5 to 2

2.Vector X and vecotor Y with size of 1M are being plugged in the equation $2X^2 + XY$ and then performed element wise multiplications and addition.

3.The vector of these 1M results is compared to 4 element-wisely. If an element in the vector is less than 4, the result of the comparision ended up with a **TRUE** value and this created another vector of the size 1M with only **TRUE** and **FALSE** values

$X, Y \sim U(-5, 2)$

```
X<- runif(1e6,min=-5,max =3)
Y<- runif(1e6,min=-5,max =3)
mean((2*X^2 + X*Y)<4)
```

```
## [1] 0.385148
```

Conclusion:

Around 38.5% of the 1M comparisions has the value **TRUE** and the other 61.5% has the value **FALSE**. In short, the probability probablity that this statement or event $2X^2 + XY < 4$ is going to occur with X and Y as uniformly distributed random variables in the interval $(-5, 3)$ is 38.5%.

2)

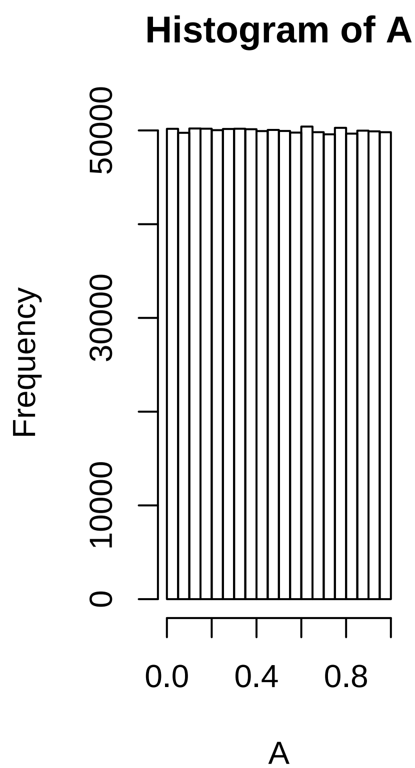
Procedure:

1.Creating two vectors X and Y that generate randomly all possible numbers within the interval $(0, 1)$

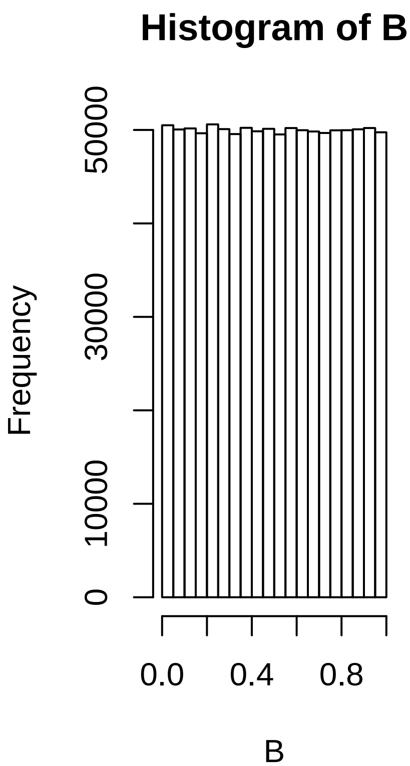
2.Ploting the histogram for the sum of two vectors

$A, B \sim U(0, 1)$

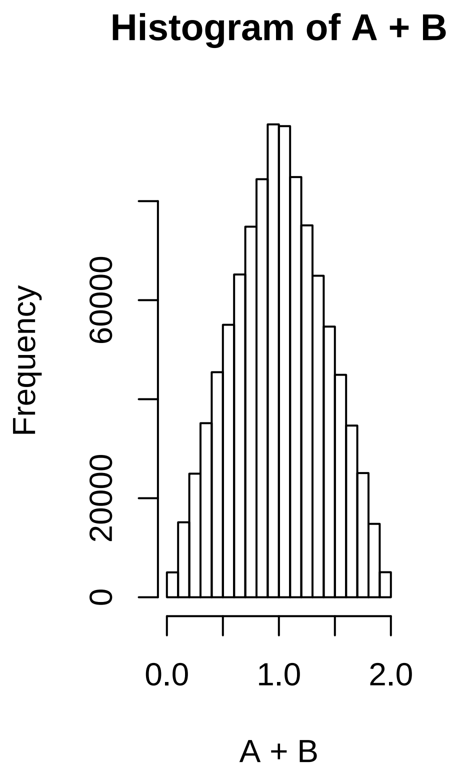
```
#logic
A<-runif(1e6, min=0,max=1)#generate rv uniformly from 0 to 1
B<-runif(1e6, min=0,max=1)
hist(A)#uniform distribution of the vector A
```




```
hist(B)#uniform distribution of the vector B
```




```r
hist(A+B)
```



Conclusion:

Each bar on the Histogram representing the number of times that a number is appeared From the uniform distribution operator $runif$. Clearly, the distribution of (A + B) does not look like a smooth stright line and this is not uniformly distributed becasue both vector A and B are random varaibles so that every possible outcome is generated with some randomness and uncertainty. Thus, the sum of two uniform distributions tends to be the expected value or mean.

3)

```r
game<-function(){
oldtoss<-(runif(1)<0.5) # flip the coin once: if TRUE we get heads, FALSE is tails
flips<-1 # counter for the number of flips until two consecutive heads
twoheads<-FALSE # a logical flag for having two consecutive heads
while(!twoheads){ # loop until two consecutive heads
newtoss<-(runif(1)<0.5) # next coin flip
flips<-flips+1 # update the flips counter
twoheads<-(oldtoss & newtoss) # update the flag for two consecutive heads
oldtoss<-newtoss # update the oldtoss for the next loop
}
return(flips) # return number of flips until two consecutive heads
}
sample<-replicate(1e4,game()) # play the game 10^4 times
mean(sample) # average number of flips until two consecutive games
```

```
## [1] 5.9802
```

4.

```
X<-runif(1e6,min=0,max=60)
Y<-runif(1e6,min=0,max=60)
mean(abs(X-Y)<5)
```

```
## [1] 0.159936
```

5.

k1: Gambler 1's initial state

k2: Gambler 2's initial state

n: Gambler plays until either $n or Ruin for k1 and k2

p: Probability of M winning $1 at each play

Function returns 1 if gambler 2 is eventually ruined

returns 0 if gambler 1 eventually wins $n

```r
gamble <- function(k1,k2,p) {
stake <- k1
stake2 <-k2
while (stake > 0 & stake2 > 0 & stake2 < (stake2+stake) & stake < (stake2+stake) ) {
bet <- sample(c(-1,1),1,prob=c(1-p,p))
stake <- stake + bet    #M
stake2 <- stake2 - bet #N
}
ifelse(stake2 == 0,return(1),return(0))
}
```

```r
k1 <- 1 #M
k2 <- 2 #N
p <- 2/3
trials <- 1e4
sample <- replicate(trials, gamble(k1,k2,p))
mean(sample) # Estimate of probability that gambler 2 is ruined
```

```
## [1] 0.5736
```

```r
4/7 #exact answer
```

```
## [1] 0.5714286
```

6.

Given: n = 1000, P(c)=0.05 mean(c)=800 rate = 1/800

E(sum(c))?

```
claim <- rbinom(1,1000,0.05)
g <-sum(rexp(claim,1/800))
x<- mean((replicate(1e6,g)))
x
```

```
## [1] 36737.92
```

P(sum(c)>50000) when policyholders = 1000?

```
#rbinom(1,1000,0.05) Generating rv for each independent trail
m1<-mean((replicate(1e5,sum(rexp(rbinom(1,1000,0.05),1/800))))>50000)#sample size of 10000
sum(rexp(claim,1/800))
```

```
## [1] 29652.5
```

```
m1
```

```
## [1] 0.10635
```

```
costClaimsPerMonth <- function(){
    pClaim = .05
    policyholders = 1000
    avgClaim = 800
    claimsPerMonth <- rbinom(1, policyholders, pClaim)
    costPerClaim <- rexp(claimsPerMonth, 1/avgClaim)
    return(sum(costPerClaim))
}

monthlyClaimCost <- replicate(1e5, costClaimsPerMonth())
mean(monthlyClaimCost)
```

```
## [1] 40004.06
```

```
targetSum = 50000
mean(monthlyClaimCost > targetSum)
```

```
## [1] 0.10579
```

7.

$Number of Claim \sim P(n, \lambda), \lambda = 10$

$Claim Amount \sim Exp(m, rate), rate = \frac{1}{1000}$

Initial capital = \$25000

Average payment per day = \$11000

Find P(Capital(t)>0) when t = 365 days?

```
mean(replicate(1e6,sum(rexp(rpois(1,10),1/1000))*365)<(25000+11000*365))
```

```
## [1] 0.633826
```

8a.

$P(s) = 3/4$

$P(f) = 1/4$

$FindE(n|s = 270) =?$ Expeted

```
Monte <- function(){
  Count<-0
  i<-0
  while(Count >= 0 & Count < 270){
    Count <- Count + sample(c(1,0),1,replace=T, prob=c(.75,0.25))#stop at 270 success
    i<-i+1
  }
  return(i)
}
```

```
mean(replicate(1000,Monte())) #around 360 times
```

```
## [1] 359.626
```

8b.

$P(X \leq x_0) = 0.99$ where X is a r.v. as the total trials

```
X<-replicate(1000,Monte())
quantile(X,.99)
```

```
## 99%
## 384
```

9.

```r
#Part 1

1-mean((replicate(1e6,rbinom(1,55,0.9))-52)<1)
```

```
## [1] 0.07716
```

```r
1-mean((rbinom(1e6,55,0.9)-52)<1)
```

```
## [1] 0.077534
```

```r
mean((rbinom(1e6,55,0.9)-52)>=1)
```

```
## [1] 0.077245
```

```r
mean((rbinom(1e6,55,0.9)-52)>0)
```

```
## [1] 0.077327
```

```r
#Part 2

#G<-the amount of gift card is been given
z<-mean((rbinom(1e6,55,0.9)-52)==1)#P(G=1)
x<-mean((rbinom(1e6,55,0.9)-52)==2)#P(G=2)
c<-mean((rbinom(1e6,55,0.9)-52)==3)#P(G=3)
mean(replicate(1e6,(1*z+2*x+3*c)*100))#E(G)
```

```
## [1] 10.2642
```

1.

Given:

Given Variables or State space:

q = The probability of moving to the left

p = The probability of moving to the Right

b = The probability of moving backward

States = { 0, 1, ,2 , 3}

a)

Probability Distribution Matrix RW:

```r
#logic for matrix RW
RW<- matrix(c(0, 1,0,0,.25,0,.75,0,0,.25,0,.75,0,0,1,0),ncol = 4,nrow = 4,byrow= TRUE)
colnames(RW) <- paste(c("0","1","2","3"))
rownames(RW) <- paste(c("0","1","2","3"))
RW
```

```
##      0    1    2    3
## 0 0.00 1.00 0.00 0.00
## 1 0.25 0.00 0.75 0.00
## 2 0.00 0.25 0.00 0.75
## 3 0.00 0.00 1.00 0.00
```

b)

$P(X_7 = 1|X_0 = 3, X_2 = 2, x_4 = 2)$

$= P(X_7 = 1|X_4 = 2)$

$= (X^3)_{21}$

$= 0.296875$

```
RW3<-matrix.power(RW,3)
RW3 #Probability distribution within 3 time steps
```

```
##          0        1        2        3
## 0 0.000000 0.437500 0.000000 0.562500
## 1 0.109375 0.000000 0.890625 0.000000
## 2 0.000000 0.296875 0.000000 0.703125
## 3 0.062500 0.000000 0.937500 0.000000
```

```
RW3[3,2]
```

```
## [1] 0.296875
```

c)

$P(X_3 = 1|X_5 = 3)$

$= \frac{P(X_5=3|X_3=1)*P(X_3=1)}{P(X_5=3)}$

$= \frac{\pi_1}{\pi_3} * P^2_{3,3}$

2.

Given Variables or State space:

S = probability of solving a HW problem successfully

F = probability of solving a HW problem unsuccessfully

States = { S,F }

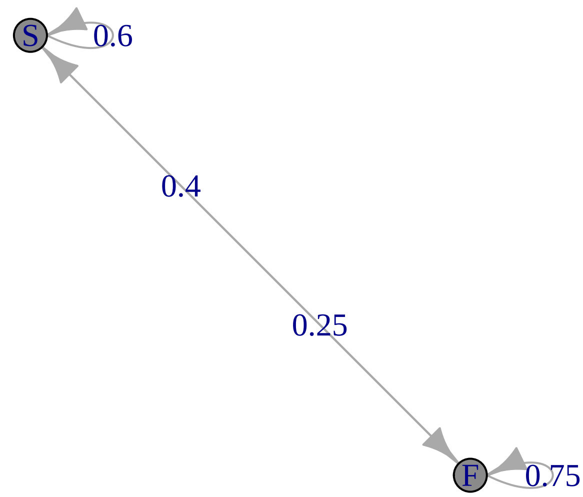
Probability Distribution Matrix HW:

```
#logic for matrix HW
HW<- matrix(c(0.6,0.4,0.25,0.75),ncol = 2,nrow = 2,byrow= TRUE)
colnames(HW) <- paste(c("S","F"))
rownames(HW) <- paste(c("S","F"))
HW
```

```
##      S    F
## S 0.60 0.40
## F 0.25 0.75
```

Relationship Diagram (HW):

```
statesNames=c("S","F")
mcB<-new("markovchain", states=statesNames, transitionMatrix=
          matrix(c(0.6,0.4,0.25,0.75),nrow=2, byrow=TRUE, dimnames=list(statesNames,
                   statesNames)
                 ))
plot(mcB)
```




```r
library(matrixcalc)
HW1<-matrix.power(HW,1)
HW16<-matrix.power(HW,16)
```

Probability distribution of success and failture in solving a HW problem at an arbitrary time:

```r
HW1 #One step transition matrix(original)
```

```
##      S    F
## S 0.60 0.40
## F 0.25 0.75
```

Probability of solving a HW successfully in a long run:

```r
HW16[,1] #Invariant distribution
```

```
##         S         F
## 0.3846154 0.3846154
```

3.

Given Variables or State space:

80 = probability of experiencing an attack From port 80

135 = probability of experiencing an attack From port 135

139 = probability of experiencing an attack From port 139

445 = probability of experiencing an attack From port 445

No attack = probability of experiencing no attack

States = { 80, 135, 139, 445 }

Time Step = 1 per week

Matrix α:

```
#logic for matrix Alpha
Alpha<- matrix(c(0,0,0,0,1),ncol = 5,nrow = 1,byrow= TRUE)
colnames(Alpha) <- paste(c("80","135","139","445", "No attack"))
rownames(Alpha) <- paste(c("Alpha"))
Alpha
```

```
##       80 135 139 445 No attack
## Alpha  0   0   0   0         1
```

Probability Distribution Matrix $HPot$:

```
#logic for matrix HPot
HPot<- matrix(c(0,0,0,0,1,0,8/13,3/13,1/13,1/13,1/16,
                3/16,3/8,1/4,1/8,0,1/11,4/11,5/11,1/11,0,1/8,1/2,1/8,1/4)
              ,ncol = 5,nrow = 5,byrow= TRUE)
colnames(HPot) <- paste(c("80","135","139","445", "No attack"))
rownames(HPot) <- paste(c("80","135","139","445", "No attack"))
HPot
```

```
##               80        135       139        445  No attack
## 80        0.0000 0.00000000 0.0000000 0.00000000 1.00000000
## 135       0.0000 0.61538462 0.2307692 0.07692308 0.07692308
## 139       0.0625 0.18750000 0.3750000 0.25000000 0.12500000
## 445       0.0000 0.09090909 0.3636364 0.45454545 0.09090909
## No attack 0.0000 0.12500000 0.5000000 0.12500000 0.25000000
```

a)

$(\alpha_k * HPot^2)_j$

```
HPot2<-matrix.power(HPot,2)
Alpha%*%HPot2
```

```
##            80       135       139       445 No attack
## Alpha 0.03125 0.2132867 0.3868007 0.2226836  0.145979
```

b)

Attack ports in long term:

```r
HPot25<-matrix.power(HPot,25)
HPot25 #Invariant distribution
```

```
##                    80       135       139       445 No attack
## 80        0.02146667 0.2669333 0.3434667 0.2273333    0.1408
## 135       0.02146667 0.2669333 0.3434667 0.2273333    0.1408
## 139       0.02146667 0.2669333 0.3434667 0.2273333    0.1408
## 445       0.02146667 0.2669333 0.3434667 0.2273333    0.1408
## No attack 0.02146667 0.2669333 0.3434667 0.2273333    0.1408
```

c)

Probability of experiencing attacks for 4 ports within 25 weeks:

```r
Alpha%*%HPot25
```

```
##                80       135       139       445 No attack
## Alpha 0.02146667 0.2669333 0.3434667 0.2273333    0.1408
```

According to the Markov Chain processs, port 139 is most likely to be attacked by hackers and part 80 is the least likely one.

4.

Given Variables or State space:

r = probability of raining

s = probability of snowing

c = probability of clear weather

States = { r, s, c }

```r
#logic for P matrix
P<- matrix(c(0.2,0.6,0.2,0.1,0.8,0.1,0.1,0.6,0.3),ncol = 3,nrow = 3,byrow= TRUE)
colnames(P) <- paste(c("r","s","c"))
rownames(P) <- paste(c("r","s","c"))


#logic for Pi matrix
Pi <- matrix(c(0.5,0.5,0),ncol=3,nrow=1,byrow = TRUE)
colnames(Pi) <- paste(c("r","s","c"))
rownames(Pi) <- paste("Pi")
```

One step transition Matrix of the weather Markov Chain (P):

```r
P
```

```
##     r   s   c
## r 0.2 0.6 0.2
## s 0.1 0.8 0.1
## c 0.1 0.6 0.3
```

```r
matrix.power(P,4)
```

```
##        r      s      c
## r 0.1112 0.7488 0.1400
## s 0.1111 0.7504 0.1385
## c 0.1111 0.7488 0.1401
```

Initial distribution (π):

```r
Pi
```

```
##      r   s c
## Pi 0.5 0.5 0
```

5.

Given Variables or State space:

Drop = probability of dropping out of college in overall

Fr = probability of dropping out of college in Freshmen level

So = probability of dropping out of college in Sophmore level

Jr = probability of dropping out of college in Junior level

Sr = probability of dropping out of college in Senior level

Grad = probability of graduating from the college in overall

States = { Drop, Fr, So, Jr, Sr, Grad }

Time step = 1 per year

```
#logic for P matrix
P<- matrix(c(1,0,0,0,0,0,.06,.03,.91,0,0,0,.06,0,.03,.91,0,0,.04,0,0,.03,.93,0,.04,0,0,0,.03,.93,0,0,0,
colnames(P) <- paste(c("Drop","Fr","So","Jr","Sr","Grad"))
rownames(P) <- paste(c("Drop","Fr","So","Jr","Sr","Grad"))


#logic for Alpha matrix
Alpha <- matrix(c(0, 1, 0, 0, 0, 0),ncol=6,nrow=1,byrow = TRUE)
colnames(Alpha) <- paste(c("Drop","Fr","So","Jr","Sr","Grad"))
rownames(Alpha) <- paste("Alpha")
```

Probability distribution of graduation rate at a perticular year(P):

```
P
```

```
##      Drop   Fr   So   Jr   Sr Grad
## Drop 1.00 0.00 0.00 0.00 0.00 0.00
## Fr   0.06 0.03 0.91 0.00 0.00 0.00
## So   0.06 0.00 0.03 0.91 0.00 0.00
## Jr   0.04 0.00 0.00 0.03 0.93 0.00
## Sr   0.04 0.00 0.00 0.00 0.03 0.93
## Grad 0.00 0.00 0.00 0.00 0.00 1.00
```

Initial distribution(α):

```
Alpha
```

```
##       Drop Fr So Jr Sr Grad
## Alpha    0  1  0  0  0    0
```

Drop out rate and graduation rate within 10 years(X_{10}):

```
P10<-matrix.power(P,10)
```

```
X10<-Alpha%*%P10
cat("Within 10 years",sep="\n")
```

```
## Within 10 years
```

```
cat("Overall drop out rate: ", X10[1,1],sep="\n")
```

```
## Overall drop out rate:
## 0.1909754
```

```
cat("Freshment drop out rate: ", X10[1,2],sep="\n")
```

```
## Freshment drop out rate:
## 5.9049e-16
```

```
cat("Sophomore drop out rate: ", X10[1,3],sep="\n")
```

```
## Sophomore drop out rate:
## 1.791153e-13
```

```
cat("Junior drop out rate: ", X10[1,4],sep="\n")
```

```
## Junior drop out rate:
## 2.444924e-11
```

```
cat("Senior drop out rate: ", X10[1,5],sep="\n")
```

```
## Senior drop out rate:
## 2.021137e-09
```

```
cat("Overall graduation rate: ", X10[1,6],sep="\n")
```

```
## Overall graduation rate:
## 0.8090246
```

Freshment graduation rate within 10 years(Fr_{10}):

```
mean(replicate(10000,(X10[1,6])))*100 #Around 81% chance
```

```
## [1] 80.90246
```