Gambler's Ruin

- Consider a gambler who starts with an initial fortune of \$1 and then on each successive gamble either wins \$1 or loses \$1 independent of the past with probabilities p and $q = 1$-p respectively.
- Suppose the gambler has a starting kitty of A.
- This gambler will place bets with the Banker, who has an initial fortune B. The Banker is, by convention, the richer of the two.
- We will look at the game from the perspective of the gambler only.

Gambler's Ruin

- Probability of successful gamble for gambler : p
- Probability of unsuccessful gamble for gambler : q (where $q = 1 - p$)
- Ratio of success probability to failure success: $s = p/q$
- Conventionally the game is biased in favour of the Banker (i.e. $q > p$ and $s < 1$)
- Total Jackpot : A+B
- Gambling concludes if Gambler , or Banker, loses everything.

Gambler's Ruin

Let R_n denote the Gamblers total fortune after the $n-$th gamble.

- If the Gambler wins the first game, his wealth becomes $R_n = A + 1$.
- If he loses the first gamble, his wealth becomes $R_n = A - 1$.
- The entire sum of money at stake is the Jackpot i.e. $A + B$.
- The game ends when the Gambler wins the Jackpot ($R_n = A + B$) or loses everything ($R_n = 0$).

Gambler's Ruin

Simulation a Single Gamble

To simulate one single bet, compute a single random number between 0 and 1.

```
runif(1)
```

- Lets assume that the game is biased in favour of the Banker p = 0.45 , q = 0.55.
- If the number is less than 0.45, the gamble wins.
- Otherwise the Banker wins.

Gambler's Ruin

```
> runif(1)
[1] 0.1251274
>#Gambler Loses
>
> runif(1)
[1] 0.754075
>#Gambler wins
>
> runif(1)
[1] 0.2132148
>#Gambler Loses
>
> runif(1)
[1] 0.8306269
```

Gambler's Ruin

The vector R_n records the gambler's worth on an ongoing basis. At the start, The first value is A.

```
A=20;B=100;p=0.47
Rn=c(A)

probval = runif(1)

if (probval < p)
 {
 A = A+1; B =B-1
 }else{A=A-1;B=B+1}

#Save the values from each bet
Rn=c(Rn,A)
```

Gambler's Ruin

Should the Gambler win the entire jackpot (A+B). The game would also cease. We include a `break` statement to stop the loop if the gambler wins the entire jackpot. A break statement will stop a loop if a certain logical condition is met.

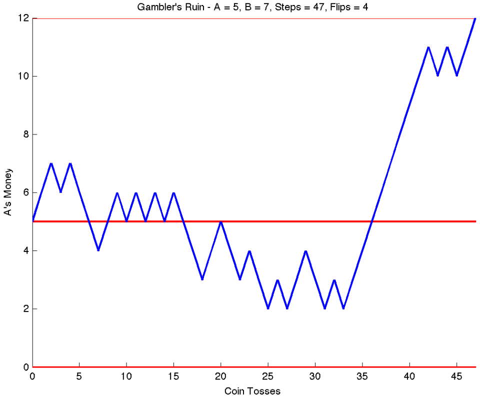

Gambler's Ruin - A = 5, B = 7, Steps = 47, Flips = 4
A's Money
Coin Tosses
0
2
4
6
8
10
12
0
5
10
15
20
25
30
35
40
45

Figure:

Gambler's Ruin

```
A=20;B=100;p=0.47
Rn=c(A)
Total=A+B

while(A>0)
  {
  UnifVal=runif(1)

  if(UnifVal <= p)
     {
     A = A+1; B =B-1
     }else{A=A-1;B=B+1}
  Rn=c(Rn,A)
  if(A==Total){break}
  }
```

Gambler's Ruin

- We can construct a plot to depict the gambler's ongoing fortunes in the game.
- We use a `for` loop, that implements the game, recording the duration of the game each time.
- The duration of each game is the dimension of the `Rn` vector, i.e. `length(Rn)`.

Gambler's Ruin

```
A.ini=20;B=100;p=0.47;M=1000
RnDist=numeric();Total=A+B
for (i in 1:M)
    {
    Rn=numeric();    Rn[1]=A;    A=A.ini
    while(A>0)
        {
        UnifVal=runif(1)

        if(UnifVal <= p)
            {
            A = A+1; B =B-1
            }else{A=A-1;B=B+1}
        Rn=c(Rn,A)
        if(A==Total){break}
```

Distribution of Durations

Simpler Approach

```
A=50;B=200;p=0.47
GAME = A+cumsum(sign(p-runif(1000)))

which(GAME==0)[1]
 # [1] 402

which(GAME==(A+B))[1]
 # [1] NA
```

Gambler's Ruin

```
T1 <- Sys.time()
A <- 50 ; B<- 100 ; p <- 0.48
M=50000
Duration = c()
GamblerWins = c()
for (i in 1:M)
{
GAME = A+cumsum(sign(p-runif(20000)))
Duration <- c(Duration, which(GAME==0)[1])
GamblerWins <- c(GamblerWins, which(GAME==(A+B))[1])
}
T2 <- Sys.time()
T2-T1
```

Gambler's Ruin

```
+ GamblerWins <- c(GamblerWins, which(GAME==(A+B))[1])
+ }
> T2 <- Sys.time()
> T2-T1
Time difference of 1.901574 mins
>
```

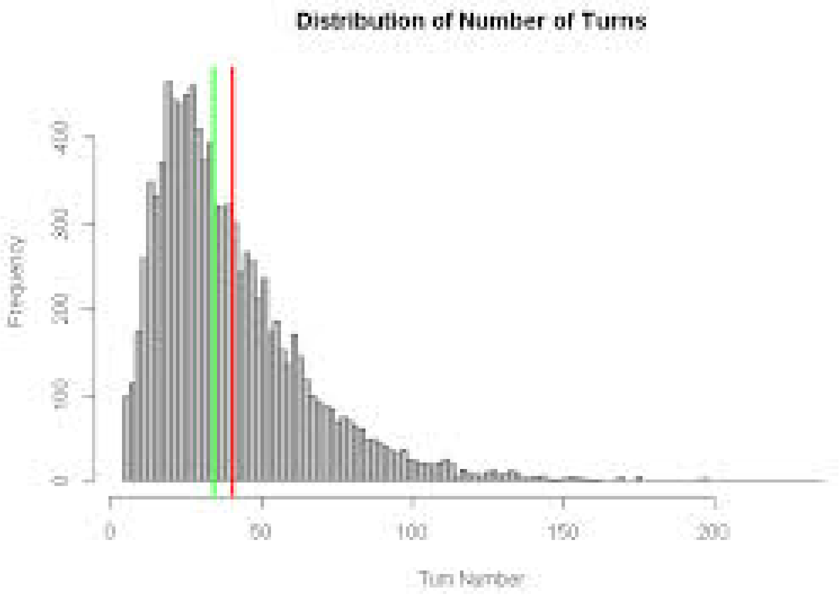
Quick enough, but could have faster implementation using Julia or RCPP.

Gambler's Ruin

Did the Gambler Ever Win?

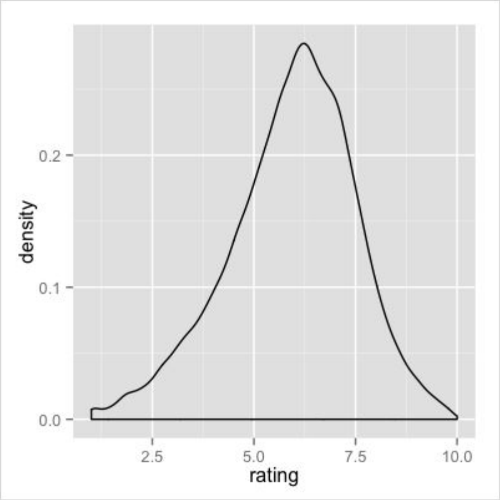
```
GamblerWins[!is.na(GamblerWins)]
which(!is.na(GamblerWins))

Keep <- which(!is.na(GamblerWins))
cbind(Duration[Keep],GamblerWins[Keep])
```

A few times! One game in every three thousand approx



Distribution of Number of Turns
Frequency
0
100
200
300
400
0
50
100
150
200
Turn Number

```
m <- ggplot(movies, aes(x = rating))
m + geom_density()
```



Kernel Density Plot