Probability with R : Gambler's Ruin

Dublin R

November 7, 2013

1 Gambler's Ruin

Consider a gambler who starts with an initial fortune of A dollars and then on each successive gamble either wins or loses independent of the past with probabilities p and $q = 1 - p$ respectively. This gamblers places bets with the Banker, who has an initial fortune of B dollars.

(For the sake of simplicity, we will look at the game from the perspective of the gambler only. The Banker is, by convention, the richer of the two, and has a better chance of winning.)

- 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 $s < 1$)

1.1 Simulation a Single Gamble

To simulate one single bet, compute a single random number between 0 and 1. For this, we use the `runif()` command, which uses generates a ***continuous uniform random variable***. The upper and lower limits can be specified. In the absence of particular specifications, the defaults are 0 and 1.

```
runif(1)
```

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

```
> runif(1)
[1] 0.1251274
```

```
>#Gambler Wins
>
> runif(1)
[1] 0.754075
>#Gambler loses
>
> runif(1)
[1] 0.2132148
>#Gambler Wins
>
> runif(1)
[1] 0.8306269
>#Gambler Loses
```

1.2 Repeated Gambles

Let A be the Gambler's Kitty at the start of the gambling. Let B be the Banker's Wealth at the outset. (Therefore the total jackpot is A+ B) The probability of gambler winning a gamble is p.

- Let Rn denote the Gamblers total fortune after the $n-$th gamble. If the Gambler wins the first game, his wealth becomes Rn = A + 1.
- If he loses the first gamble, his wealth becomes Rn = 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 (Rn = A + B) or loses everything (Rn = 0).
- The vector ***Rec*** records the gambler's worth on an ongoing basis, with the first element being the value of A. As the gambling progresses, the new value gets added to the vector.

Lets see how this plays out over a night of gambling. Lets start the gambler with 20 dollars, and the banker with 100. Unknown to the gambler is the fact that he or she has only a 47% chance of winning a single bet.

```
#initial values
A=20;B=100;p=0.47

#record the progress of the gambler
Rec=c(A)

#generate a outcome for each gamble.
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
Rec=c(Rec,A)
```

1.3 Using a `while` loop

So far we have managed to code for a single bet. Lets simulate the gambling in its entirety. For this, we will use a `while` loop. A a `while` loop will repeat a set of commands as long as a particular logical condition is met. (Here, that logical condition is that the gambler has money to gamble with.i.e. $A > 0$). Once he loses everything, the while loop (and the gambling) terminates.

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

while(A>0)
{
    ProbVal=runif(1)
    if(ProbVal <= p)
      {
      A = A+1; B =B-1
      }else{A=A-1;B=B+1}
    Rec=c(Rec,A)

}
```

We will also include a break statement to stop the loop in the unlikely event that the Gambler wins the jackpot.

```
A=20;B=100;p=0.47
Rec=c(A)
Total=A+B  #total jackpot

while(A>0)
   {
   ProbVal=runif(1)
   if(ProbVal <= p)
   {
   A = A+1; B =B-1
   }else{A=A-1;B=B+1}
   Rec=c(Rec,A)
   if(A==Total){break}
   }
```

1.4 The Gambler's Performance

In all likelihood, the Gambler lost everything. But how long did he last in the game? how many gambles was he able to play? Also What was his highest level of wealth? We can look at the ***Rec*** vector to answer these questions.

```
length(Rec)
max(Rec)
```

We can construct a plot to depict the gambler's ongoing fortunes in the game.

```
# Line plot, colour red
plot(Rec,type="l",col="red")

#Axes
abline(h=0)
abline(v=0)

#Also the gamblers initial value and the total value of the game(optional)
abline(h=A,col="red")
abline(h=Total,col="green")
```

1.5 Simulation Study

Suppose we are interested in the probability distribution of durations. What is the probability of lasting more than 100 rounds? less than 200 etc? We can use a function called `GambRuinFunc()` to compute the durations from an number of nights of gambling (next page) . The function can easily be adjusted to consider the maximum value of the gambler's worth We can specify values for A , B and p different to that of the defaults. Here, they are picked so as to be quick to execute.

```
Durations= numeric()
M=1000
for(i in 1:M)
   {
   NextDuration = GambRuinFunc(A=10,B=30,p=0.40)
   Durations=c(Durations,NextDuration)
   }
```

We can study the Durations vector using simple statistical functions.

```
hist(Durations)
mean(Durations)
median(Durations)
IQR(Durations)
quantile(Durations,1:20/20)
```

```
GambRuinFunc=function(A=20,B=100,p=0.47)
{
Rec=c(A)
Total=A+B  #total jackpot

while(A>0)
   {
   ProbVal=runif(1)
   if(ProbVal <= p)
   {
   A = A+1; B =B-1
   }else{A=A-1;B=B+1}
   Rec=c(Rec,A)
   if(A==Total){break}
   }
Duration=length(Rec)
return(Duration)
}
```

```
> quantile(DurDist)
  0%  25%  50%  75% 100%
  23  139  239  419 4167
>
> quantile(DurDist,1:10/10)
 10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
  91  123  157  195  239  295  369  481  691 4167
```

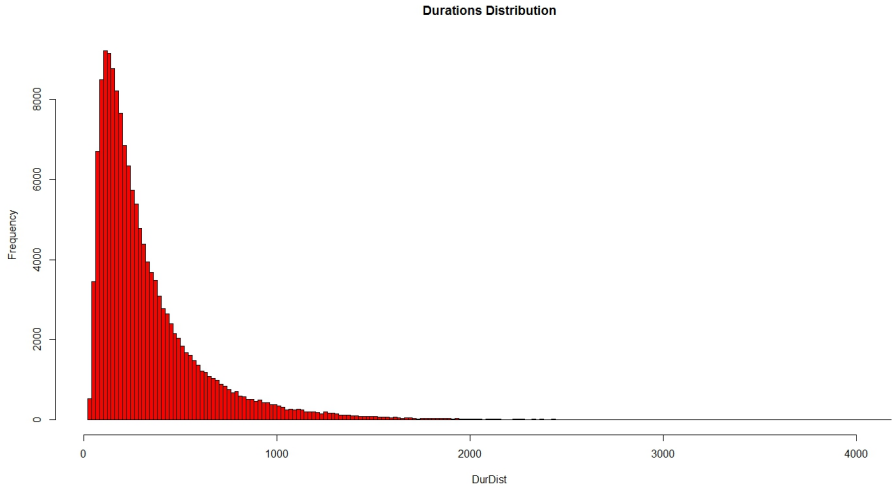

Figure 1

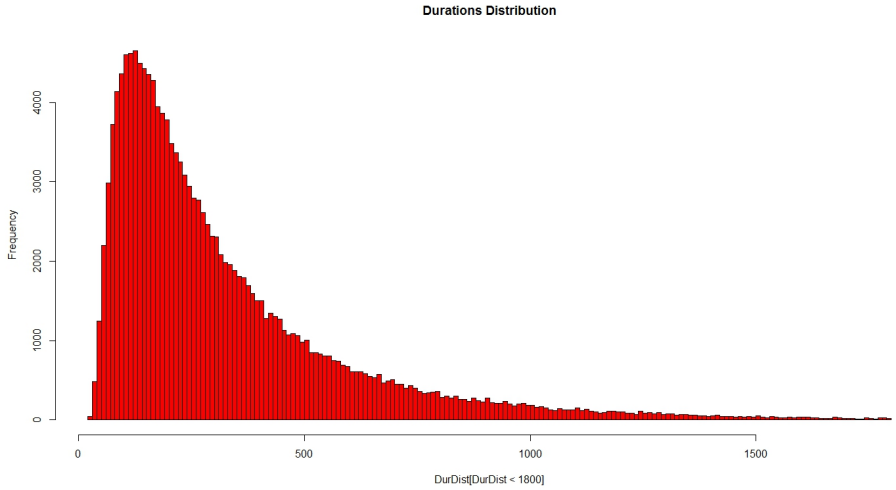

Figure 2