Slot_Machine

Saurav Singh

17 January 2017

Slot Machine

In this Assignment I have build a real working slot machine modelled after real life Video Lottery Terminal from Manitoba,Canada.
Firstly, Defining a function get_symbols() which with helds all the symbols of slot machine with weighted probabilities and returns 3 randomed symbols.

```
get_symbols<- function(){
  wheel<- c("DD","7","BBB","BB","B","C","0")
  sample(wheel,size = 3,replace = TRUE,prob = c(0.03,0.03,0.06,0.1,0.25,0.01,0.52))
}
get_symbols()
```

```
## [1] "0" "0" "0"
```

Next created a score() function which doesnt considers the DD-double diamonds as Wildcards:

```
score<-function(symbols) {
same<- symbols[1]==symbols[2] && symbols[2]==symbols[3]
bars<-symbols %in% c("B","BB","BBB")
if(same) {
  payouts<-c("DD"=100,"7"=80,"BBB"=40,"BB"=25,"B"=10,"C"=10,"0"=0)
  prize<- unname(payouts[symbols[1]])
} else if(all(bars)) {
  prize<- 5
} else {
  cherries<-sum(symbols=="C")
  prize<- c(0,2,5)[cherries+1]
}
diamonds<-sum(symbols=="DD")
prize*2^diamonds
}
```

Next created a score1() function which considers the real scenario slot machine considers DD-double diamonds as Wildcards:

```
score1 <- function(symbols) {
  diamonds <- sum(symbols == "DD")
  cherries <- sum(symbols == "C")
  slots <- symbols[symbols != "DD"]
  same <- length(unique(slots)) == 1
  bars <- slots %in% c("B", "BB", "BBB")
  if (diamonds == 3) {
    prize <- 100
  } else if (same) {
    payouts <- c("7" = 80, "BBB" = 40, "BB" = 25,
                 "B" = 10, "C" = 10, "0" = 0)
    prize <- unname(payouts[slots[1]])
  } else if (all(bars)) {
```

```r
    prize <- 5
  } else if (cherries > 0) {
    prize <- c(0, 2, 5)[cherries + diamonds + 1]
  } else {
    prize <- 0
  }
  prize * 2^diamonds
}
```

Next,I have created slot_display function which will show the out of play() in a fashion manner withour quotes.Further i have overloaded print method so as to call slot_display().

```r
slot_display<- function(prize) {
  symbols<-attr(prize,"symbols")
  symbols<-paste(symbols,collapse = " ")
  string<-paste(symbols,prize,sep="\n$")
  cat(string)
}

print.slots<-function(x,...){
  slot_display(x)
}
```

Next created a play() function which we will use for playing our slot machine.

```r
play<-function(){
  symbols<-get_symbols()
  structure(score1(symbols),symbols=symbols,class="slots")
}
play()
```

```
## BB 0 0
## $0
```

Expected Value

Calculated Expected Value for 1st case i.e by not considering DD-double diamonds as Wildcards

```r
wheel<- c("DD","7","BBB","BB","B","C","0")
combos<-expand.grid(wheel,wheel,wheel,stringsAsFactors = FALSE)
prob<-c("DD"=0.03,"7"=0.03,"BBB"=0.06,"BB"=0.1,"B"=0.25,"C"=0.01,"0"=0.52)
combos$prob1<-prob[combos$Var1]
combos$prob2<-prob[combos$Var2]
combos$prob3<-prob[combos$Var3]
combos$prob<-combos$prob1*combos$prob2*combos$prob3
combos$prize<-NA
for(i in 1:nrow(combos)){
  symbols<-c(combos[i,1],combos[i,2],combos[i,3])
  combos$prize[i]<-score(symbols)
}
sum(combos$prize*combos$prob)
```

```
## [1] 0.538014
```

Now,Calculating for the real case i.e by considering DD-double diamonds as Wildcards.

```
for(i in 1:nrow(combos)){
  symbols<-c(combos[i,1],combos[i,2],combos[i,3])
  combos$prize[i]<-score1(symbols)
}
sum(combos$prize*combos$prob)
```

```
## [1] 0.934356
```

Learning about While loops

The above example tells us about how to use While loops in R.
(the function basically takes a value(i.e money you have before you started playing) as an argument and then we play on slot machine unless our money gets over, it returns how many times we played in this period of segment.)

```
plays_till_broke <- function(start_with) {
  cash <- start_with
  n <- 0
  while (cash > 0) {
    cash <- cash - 1 + play()
    n <- n + 1
  }
  n
}
plays_till_broke(100)
```

```
## [1] 192
```

Learning about Repeat loops

The above example tells us about how to use Repeat loops in R.
(funtionality same as above)

```
plays_till_broke1 <- function(start_with) {
  cash <- start_with
  n <- 0
  repeat {
    cash <- cash - 1 + play()
    n <- n + 1
    if (cash <= 0) {
      break
    }
  }
  n
}
plays_till_broke1(100)
```

```
## [1] 555
```

Vectorized codes

Vectorized codes are usually faster than loops.It uses the combination of logical subsetting with vectors.The fastest R codes generally have 3 things in common:logical tests,susetting and element-wise execution.

Let us compare the execution time:
For loops:

```
abs_loop<- function(vec){
  for(i in 1:length(vec)){
    if(vec[i]<0){
     vec[i]<- -vec[i]
    }
  }
  vec
}
long<-rep(c(-1,1),5000000)
system.time(abs_loop(long))
```

```
##    user  system elapsed
##   9.172   0.100   9.282
```

For vectorized code:

```
abs_sets<- function(vec){
  negs<-vec<0
  vec[negs]<-vec[negs]*-1
  vec
}
long<-rep(c(-1,1),5000000)
system.time(abs_sets(long))
```

```
##    user  system elapsed
##   0.312   0.104   0.416
```

As, we can see how fast a vectorized code can execute.
Let's use it to our advantage by calculating the mean of winnings by calling play() function a million times.

```
winnings<-vector(length=1000000)
for(i in 1:1000000){
  winnings[i]<-play()
}
mean(winnings)
```

```
## [1] 0.944608
```