Project-1

Yash Verma

31 August 2016

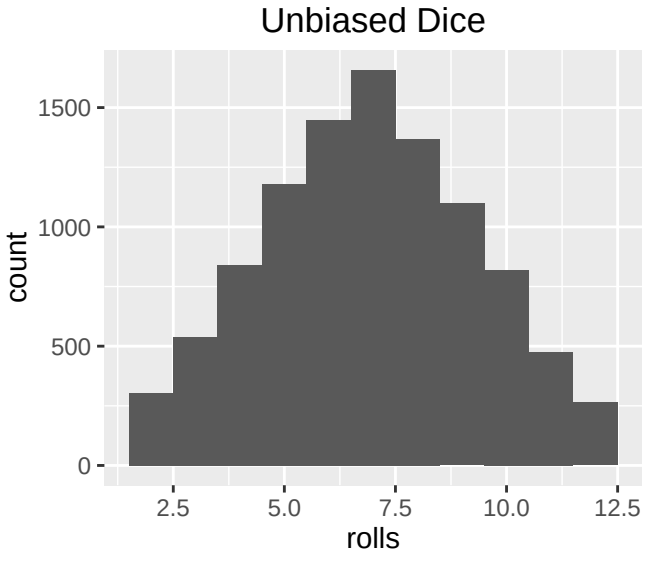
Rolling a dice

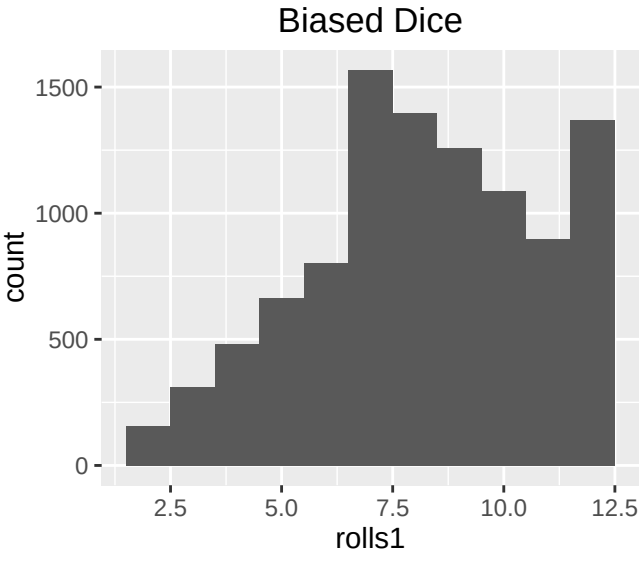
Code for a normal Dice:

```
roll<-function(){
  die<-1:6
  dice<-sample(die,size=2,replace=TRUE)
  sum(dice)
}

rolls<-replicate(10000,roll())
```

Code for a weighted Dice:

```
roll1<-function(){
  die<-1:6
  dice<-sample(die,size=2,replace=TRUE,prob<-c(1/8,1/8,1/8,1/8,1/8,3/8))
  sum(dice)
}

rolls1<-replicate(10000,roll1())
```





A deck of Cards

Code for a deck made using a data frame

```
deck <- data.frame(
  face = c("king", "queen", "jack", "ten", "nine", "eight", "seven", "six",
           "five", "four", "three", "two", "ace", "king", "queen", "jack", "ten",
           "nine", "eight", "seven", "six", "five", "four", "three", "two", "ace",
           "king", "queen", "jack", "ten", "nine", "eight", "seven", "six", "five",
           "four", "three", "two", "ace", "king", "queen", "jack", "ten", "nine",
           "eight", "seven", "six", "five", "four", "three", "two", "ace"),
  suit = c("spades", "spades", "spades", "spades", "spades", "spades",
           "spades", "spades", "spades", "spades", "spades", "spades", "spades",
           "clubs", "clubs", "clubs", "clubs", "clubs", "clubs", "clubs", "clubs",
           "clubs", "clubs", "clubs", "clubs", "clubs", "diamonds", "diamonds",
           "diamonds", "diamonds", "diamonds", "diamonds", "diamonds", "diamonds",
           "diamonds", "diamonds", "diamonds", "diamonds", "diamonds", "hearts",
           "hearts", "hearts", "hearts", "hearts", "hearts", "hearts", "hearts",
           "hearts", "hearts", "hearts", "hearts", "hearts"),
  value = c(13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 13, 12, 11, 10, 9, 8,
            7, 6, 5, 4, 3, 2, 1, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 13, 12, 11,
            10, 9, 8, 7, 6, 5, 4, 3, 2, 1)
)
```

Code for dealing a Card from the Deck

```
deal <- function(deck){
  deck[1,]
}

deal(deck)
```

```
##   face   suit value
## 1 king spades    13
```

Code for shuffling a deck

```
shuffle<-function(deck){
  random<-sample(1:52,size=52)
  deck[random, ]
}
```

Changing the value of a particular card in a shuffled deck

```
deck2 <- shuffle(deck)
deck2$value[deck2$face=="ace"]<-14
```

To remove the 'Not Available' data while calculations, we use the `na.rm = TRUE`

Real life card dealing

```
deal<-function(){
  card<-deck[1,]
  assign("deck",deck[-1,],envir=globalenv())
  card
}
```

Real life shuffling

```
shuffle<-function(){
  random <- sample(1:52,size=52)
  assign("deck",deck[random, ],envir = globalenv())
}
```

Keeping original deck safe/Closure

```
setup <- function(deck) {
DECK <- deck
DEAL <- function() {
card <- deck[1, ]
assign("deck", deck[-1, ], envir = parent.env(environment()))
card
}
SHUFFLE <- function(){
random <- sample(1:52, size = 52)
assign("deck", DECK[random, ], envir = parent.env(environment()))
}
list(deal = DEAL, shuffle = SHUFFLE)
}

cards <- setup(deck)
deal <- cards$deal
shuffle <- cards$shuffle
```

Closure ensures that even if we remove the original deck, we can continue playing cards

Slot Machine

A code in R which allows us to play the most popular mordern casino game.
Symbols used include the following :
DD - Diamonds (0.03)
7 - Seven (0.03)
BBB - Triple Bars (0.06)
BB - Double Bars (0.1)
B - Single Bars (0.25)
C - Cherries (0.01)
0 - Zeros (0.52)
with the probabilities in the brackets.
Symbols are selected randomly using the sample function and the code is as follows:

```
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] "B" "B" "0"
```

For the actual game, we need to assign score to the symbols and that can be done via the score function. The score of the 3 random symbols obtained from 'get_symbols' is extracted from a lookup table which has all the values.
The code is as follows :

```
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
}
```

The game can be run using this function which calls the get_symbols() functions and the score() function.

```
play <- function(){
  symbols <- get_symbols()
  print(symbols)
  score(symbols)
}
play()
```

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

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

Modified play() function to store the values of the symbols as attributes with the value of prize.

```
play <- function(){
  symbols <- get_symbols()
  structure(score(symbols),symbols = symbols)
}
play()
```

```
## [1] 0
## attr(,"symbols")
## [1] "0" "B" "0"
```

The structure function creates an object with a set of attributes. The first argument should be a R object or set of values and the remaining arguments should be named attributes for the structure to add to the object. The attributes now can be used to create a slot_display() functiona as follows:

```
slot_display <- function(prize){
  #extract the symbols
  symbols <- attr(prize,"symbols")
  #combine symbol with prize as a regular expression
  symbols <- paste(symbols,collapse = " ")
  #append with new line character
  string <- paste(symbols,prize,sep="\n$")
  cat(string)   #display without quotes
}
one_play <- play()
```

We use expand.grid to find out all the possible combinations of a vector with another vector. Using this we calculate the possible combinations of the wheel.

```
wheel <- c("DD","7","BBB","BB","B","C","0")
combos <- expand.grid(wheel,wheel,wheel,stringsAsFactors = FALSE)
head(combos,3)
```

```
##   Var1 Var2 Var3
## 1   DD   DD   DD
## 2    7   DD   DD
## 3  BBB   DD   DD
```

This creates a variable combos with 343 observations.
We then create a new lookup table for the probabilites and add those values to combos as a factor and, calculate and add a total probability for each of the combination.

```
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
head(combos,3)
```

```
##   Var1 Var2 Var3 prob1 prob2 prob3    prob
## 1   DD   DD   DD  0.03  0.03  0.03 2.7e-05
## 2    7   DD   DD  0.03  0.03  0.03 2.7e-05
## 3  BBB   DD   DD  0.06  0.03  0.03 5.4e-05
```

To store the prize along with the combinations, we use a for loop and add prize as a factor.

```
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)
}
head(combos,3)
```

```
##   Var1 Var2 Var3 prob1 prob2 prob3    prob prize
## 1   DD   DD   DD  0.03  0.03  0.03 2.7e-05   800
## 2    7   DD   DD  0.03  0.03  0.03 2.7e-05     0
## 3  BBB   DD   DD  0.06  0.03  0.03 5.4e-05     0
```

The expected value of the prize won is given by

```
sum(combos$prize * combos$prob)
```

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

This value is less than the value mentioned by the manufacturer because of the wild card “DD”.
Correcting the code to fix the wild card problem and recalculating the expected value of prize.

```
score <- function(symbols){
  diamonds <- sum(symbols == "DD")
  cherries <- sum(symbols == "C")
  #case identification
  slots <- symbols[symbols != "DD"]
  same <- length(unique(slots)) == 1
  bars <- slots %in% c("B","BB","BBB")
  #assign prize
  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)){
    prize <- 5
  } else if(cherries > 0){
    prize <- c(0,2,5)[cherries + diamonds + 1]
  } else {
    prize <- 0
  }
  #double the prize for each diamond
  prize * 2^diamonds
}

#reassinging the prize value
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)
}

#finding sum
sum(combos$prize * combos$prob)
```

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

To demonstrate the playing in till the cash runs out

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