Rest of Book

Gavin McCorry

2024-04-25

Code From Chap 7

```
get_symbols <- function(){
  wheel <- c("DD", "7", "BBB", "BB", "B", "C", "0")
  sample(wheel, size = 3, replace = TRUE,
         prob = c(.03, .03, .06, .1, .25, .01, .52))
}

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
}

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

Chapter 8: S3

```
# adding attributes to objects
one_play <- play()
```

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

```
one_play
```

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

```
attributes(one_play)
```

```
## NULL
```

```
# Right way to do it
attr(one_play, "symbols") <- c("B", "0", "B")

attributes(one_play)
```

```
## $symbols
## [1] "B" "0" "B"
```

```
# to look u the value of any attribue
attr(one_play, "symbols")
```

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

```
# R displays the attriubue beeneath the vectors values
one_play
```

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

Exercise 10.1: (Add an Attribute) Modify play to return a prize that contains the symbols associated with it as an attribute named symbols. Remove the redundant call to print(symbols):

```
play <- function() {
  symbols <- get_symbols()
  prize <- score(symbols)
  attr(prize, "symbols") <- symbols
  prize
}

# play now returns both the prize and th symbols associatd with the prize
play()
```

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

```
two_play <- play()

two_play
```

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

slot display function:

```
slot_display <- function(prize){

  # extract symbols
  symbols <- attr(prize, "symbols")

  # collapse symbols into single string
  symbols <- paste(symbols, collapse = " ")

  # combine symbol with prize as a character string
  # \n is special escape sequence for a new line (i.e. return or enter)
  string <- paste(symbols, prize, sep = "\n$")

  # display character string in console without quotes
  cat(string)
}

slot_display(one_play)
```

```
## B 0 B
## $0
```

```
slot_display(play())
```

```
## 0 DD B
## $0
```

Methods

```
# givign on_pla a class
class(one_play) <- "slots"

# Writing a print method for our new class
args(print)
```

```
## function (x, ...)
## NULL
```

```
## function (x, ...)
## NULL

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

# Adding classes to the play function
play <- function() {
  symbols <- get_symbols()
```

```
  structure(score(symbols), symbols = symbols, class = "slots")
}

class(play())
```

```
## [1] "slots"
```

```
play()
```

```
## 0 B 0
## $0
```

Chapter 9: Loops

```
die <- c(1, 2, 3, 4, 5, 6)
```

Expand.grid

```
# this function in R provides a uick way to write out every combinatin of the elements in n vectors.
rolls <- expand.grid(die, die)
rolls
```

```
##    Var1 Var2
## 1     1    1
## 2     2    1
## 3     3    1
## 4     4    1
## 5     5    1
## 6     6    1
## 7     1    2
## 8     2    2
## 9     3    2
## 10    4    2
## 11    5    2
## 12    6    2
## 13    1    3
## 14    2    3
## 15    3    3
## 16    4    3
## 17    5    3
## 18    6    3
## 19    1    4
## 20    2    4
## 21    3    4
## 22    4    4
## 23    5    4
## 24    6    4
```

```
## 25    1    5
## 26    2    5
## 27    3    5
## 28    4    5
## 29    5    5
## 30    6    5
## 31    1    6
## 32    2    6
## 33    3    6
## 34    4    6
## 35    5    6
## 36    6    6
```

```
# You can determine the value of each roll once you've made your list of outcomes. This will be the sum
rolls$value <- rolls$Var1 + rolls$Var2
head(rolls, 3)
```

```
##   Var1 Var2 value
## 1    1    1     2
## 2    2    1     3
## 3    3    1     4
```

```
# lookup table for probabilitues of rollings valeus in var1
prob <- c("1" = 1/8, "2" = 1/8, "3" = 1/8, "4" = 1/8, "5" = 1/8, "6" = 3/8)
prob
```

```
##     1     2     3     4     5     6
## 0.125 0.125 0.125 0.125 0.125 0.375
```

```
# can subset rolls$var1 to get a vector of probabilities
rolls$Var1
```

```
##  [1] 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6
```

```
prob[rolls$Var1]
```

```
##     1     2     3     4     5     6     1     2     3     4     5     6     1
## 0.125 0.125 0.125 0.125 0.125 0.375 0.125 0.125 0.125 0.125 0.125 0.375 0.125
##     2     3     4     5     6     1     2     3     4     5     6     1     2
## 0.125 0.125 0.125 0.125 0.375 0.125 0.125 0.125 0.125 0.125 0.375 0.125 0.125
##     3     4     5     6     1     2     3     4     5     6
## 0.125 0.125 0.125 0.375 0.125 0.125 0.125 0.125 0.125 0.375
```

```
rolls$prob1 <- prob[rolls$Var1]
head(rolls, 3)
```

```
##   Var1 Var2 value prob1
## 1    1    1     2 0.125
## 2    2    1     3 0.125
## 3    3    1     4 0.125
```

```r
# can do the sam far var2
rolls$prob2 <- prob[rolls$Var2]
head(rolls, 3)
```

```
##   Var1 Var2 value prob1 prob2
## 1    1    1     2 0.125 0.125
## 2    2    1     3 0.125 0.125
## 3    3    1     4 0.125 0.125
```

```r
# can calculat the probability of rolling each combination:
rolls$prob <- rolls$prob1 * rolls$prob2
head(rolls, 3)
```

```
##   Var1 Var2 value prob1 prob2     prob
## 1    1    1     2 0.125 0.125 0.015625
## 2    2    1     3 0.125 0.125 0.015625
## 3    3    1     4 0.125 0.125 0.015625
```

```r
# expected value
sum(rolls$value * rolls$prob)
```

```
## [1] 8.25
```

Calculating the expectd roll of the slot machine

```r
# dfine wheel
wheel <- c("DD", "7", "BBB", "BB", "B", "C", "0")

# All combinations
combos <- expand.grid(wheel, wheel, wheel, stringsAsFactors = FALSE)
combos
```

```
##     Var1 Var2 Var3
## 1     DD   DD   DD
## 2      7   DD   DD
## 3    BBB   DD   DD
## 4     BB   DD   DD
## 5      B   DD   DD
## 6      C   DD   DD
## 7      0   DD   DD
## 8     DD    7   DD
## 9      7    7   DD
## 10   BBB    7   DD
## 11    BB    7   DD
## 12     B    7   DD
## 13     C    7   DD
## 14     0    7   DD
## 15    DD  BBB   DD
## 16     7  BBB   DD
## 17   BBB  BBB   DD
```

```
## 18    BB  BBB   DD
## 19     B  BBB   DD
## 20     C  BBB   DD
## 21     0  BBB   DD
## 22    DD   BB   DD
## 23     7   BB   DD
## 24   BBB   BB   DD
## 25    BB   BB   DD
## 26     B   BB   DD
## 27     C   BB   DD
## 28     0   BB   DD
## 29    DD    B   DD
## 30     7    B   DD
## 31   BBB    B   DD
## 32    BB    B   DD
## 33     B    B   DD
## 34     C    B   DD
## 35     0    B   DD
## 36    DD    C   DD
## 37     7    C   DD
## 38   BBB    C   DD
## 39    BB    C   DD
## 40     B    C   DD
## 41     C    C   DD
## 42     0    C   DD
## 43    DD    0   DD
## 44     7    0   DD
## 45   BBB    0   DD
## 46    BB    0   DD
## 47     B    0   DD
## 48     C    0   DD
## 49     0    0   DD
## 50    DD   DD    7
## 51     7   DD    7
## 52   BBB   DD    7
## 53    BB   DD    7
## 54     B   DD    7
## 55     C   DD    7
## 56     0   DD    7
## 57    DD    7    7
## 58     7    7    7
## 59   BBB    7    7
## 60    BB    7    7
## 61     B    7    7
## 62     C    7    7
## 63     0    7    7
## 64    DD  BBB    7
## 65     7  BBB    7
## 66   BBB  BBB    7
## 67    BB  BBB    7
## 68     B  BBB    7
## 69     C  BBB    7
## 70     0  BBB    7
## 71    DD   BB    7
```

```
## 72     7   BB    7
## 73   BBB   BB    7
## 74    BB   BB    7
## 75     B   BB    7
## 76     C   BB    7
## 77     0   BB    7
## 78    DD    B    7
## 79     7    B    7
## 80   BBB    B    7
## 81    BB    B    7
## 82     B    B    7
## 83     C    B    7
## 84     0    B    7
## 85    DD    C    7
## 86     7    C    7
## 87   BBB    C    7
## 88    BB    C    7
## 89     B    C    7
## 90     C    C    7
## 91     0    C    7
## 92    DD    0    7
## 93     7    0    7
## 94   BBB    0    7
## 95    BB    0    7
## 96     B    0    7
## 97     C    0    7
## 98     0    0    7
## 99    DD   DD  BBB
## 100    7   DD  BBB
## 101  BBB   DD  BBB
## 102   BB   DD  BBB
## 103    B   DD  BBB
## 104    C   DD  BBB
## 105    0   DD  BBB
## 106   DD    7  BBB
## 107    7    7  BBB
## 108  BBB    7  BBB
## 109   BB    7  BBB
## 110    B    7  BBB
## 111    C    7  BBB
## 112    0    7  BBB
## 113   DD  BBB  BBB
## 114    7  BBB  BBB
## 115  BBB  BBB  BBB
## 116   BB  BBB  BBB
## 117    B  BBB  BBB
## 118    C  BBB  BBB
## 119    0  BBB  BBB
## 120   DD   BB  BBB
## 121    7   BB  BBB
## 122  BBB   BB  BBB
## 123   BB   BB  BBB
## 124    B   BB  BBB
## 125    C   BB  BBB
```

```
## 126    0   BB  BBB
## 127   DD    B  BBB
## 128    7    B  BBB
## 129  BBB    B  BBB
## 130   BB    B  BBB
## 131    B    B  BBB
## 132    C    B  BBB
## 133    0    B  BBB
## 134   DD    C  BBB
## 135    7    C  BBB
## 136  BBB    C  BBB
## 137   BB    C  BBB
## 138    B    C  BBB
## 139    C    C  BBB
## 140    0    C  BBB
## 141   DD    0  BBB
## 142    7    0  BBB
## 143  BBB    0  BBB
## 144   BB    0  BBB
## 145    B    0  BBB
## 146    C    0  BBB
## 147    0    0  BBB
## 148   DD   DD   BB
## 149    7   DD   BB
## 150  BBB   DD   BB
## 151   BB   DD   BB
## 152    B   DD   BB
## 153    C   DD   BB
## 154    0   DD   BB
## 155   DD    7   BB
## 156    7    7   BB
## 157  BBB    7   BB
## 158   BB    7   BB
## 159    B    7   BB
## 160    C    7   BB
## 161    0    7   BB
## 162   DD  BBB   BB
## 163    7  BBB   BB
## 164  BBB  BBB   BB
## 165   BB  BBB   BB
## 166    B  BBB   BB
## 167    C  BBB   BB
## 168    0  BBB   BB
## 169   DD   BB   BB
## 170    7   BB   BB
## 171  BBB   BB   BB
## 172   BB   BB   BB
## 173    B   BB   BB
## 174    C   BB   BB
## 175    0   BB   BB
## 176   DD    B   BB
## 177    7    B   BB
## 178  BBB    B   BB
## 179   BB    B   BB
```

```
## 180    B    B   BB
## 181    C    B   BB
## 182    0    B   BB
## 183   DD    C   BB
## 184    7    C   BB
## 185  BBB    C   BB
## 186   BB    C   BB
## 187    B    C   BB
## 188    C    C   BB
## 189    0    C   BB
## 190   DD    0   BB
## 191    7    0   BB
## 192  BBB    0   BB
## 193   BB    0   BB
## 194    B    0   BB
## 195    C    0   BB
## 196    0    0   BB
## 197   DD   DD    B
## 198    7   DD    B
## 199  BBB   DD    B
## 200   BB   DD    B
## 201    B   DD    B
## 202    C   DD    B
## 203    0   DD    B
## 204   DD    7    B
## 205    7    7    B
## 206  BBB    7    B
## 207   BB    7    B
## 208    B    7    B
## 209    C    7    B
## 210    0    7    B
## 211   DD  BBB    B
## 212    7  BBB    B
## 213  BBB  BBB    B
## 214   BB  BBB    B
## 215    B  BBB    B
## 216    C  BBB    B
## 217    0  BBB    B
## 218   DD   BB    B
## 219    7   BB    B
## 220  BBB   BB    B
## 221   BB   BB    B
## 222    B   BB    B
## 223    C   BB    B
## 224    0   BB    B
## 225   DD    B    B
## 226    7    B    B
## 227  BBB    B    B
## 228   BB    B    B
## 229    B    B    B
## 230    C    B    B
## 231    0    B    B
## 232   DD    C    B
## 233    7    C    B
```

```
## 234  BBB    C    B
## 235   BB    C    B
## 236    B    C    B
## 237    C    C    B
## 238    0    C    B
## 239   DD    0    B
## 240    7    0    B
## 241  BBB    0    B
## 242   BB    0    B
## 243    B    0    B
## 244    C    0    B
## 245    0    0    B
## 246   DD   DD    C
## 247    7   DD    C
## 248  BBB   DD    C
## 249   BB   DD    C
## 250    B   DD    C
## 251    C   DD    C
## 252    0   DD    C
## 253   DD    7    C
## 254    7    7    C
## 255  BBB    7    C
## 256   BB    7    C
## 257    B    7    C
## 258    C    7    C
## 259    0    7    C
## 260   DD  BBB    C
## 261    7  BBB    C
## 262  BBB  BBB    C
## 263   BB  BBB    C
## 264    B  BBB    C
## 265    C  BBB    C
## 266    0  BBB    C
## 267   DD   BB    C
## 268    7   BB    C
## 269  BBB   BB    C
## 270   BB   BB    C
## 271    B   BB    C
## 272    C   BB    C
## 273    0   BB    C
## 274   DD    B    C
## 275    7    B    C
## 276  BBB    B    C
## 277   BB    B    C
## 278    B    B    C
## 279    C    B    C
## 280    0    B    C
## 281   DD    C    C
## 282    7    C    C
## 283  BBB    C    C
## 284   BB    C    C
## 285    B    C    C
## 286    C    C    C
## 287    0    C    C
```

```
## 288   DD    0    C
## 289    7    0    C
## 290  BBB    0    C
## 291   BB    0    C
## 292    B    0    C
## 293    C    0    C
## 294    0    0    C
## 295   DD   DD    0
## 296    7   DD    0
## 297  BBB   DD    0
## 298   BB   DD    0
## 299    B   DD    0
## 300    C   DD    0
## 301    0   DD    0
## 302   DD    7    0
## 303    7    7    0
## 304  BBB    7    0
## 305   BB    7    0
## 306    B    7    0
## 307    C    7    0
## 308    0    7    0
## 309   DD  BBB    0
## 310    7  BBB    0
## 311  BBB  BBB    0
## 312   BB  BBB    0
## 313    B  BBB    0
## 314    C  BBB    0
## 315    0  BBB    0
## 316   DD   BB    0
## 317    7   BB    0
## 318  BBB   BB    0
## 319   BB   BB    0
## 320    B   BB    0
## 321    C   BB    0
## 322    0   BB    0
## 323   DD    B    0
## 324    7    B    0
## 325  BBB    B    0
## 326   BB    B    0
## 327    B    B    0
## 328    C    B    0
## 329    0    B    0
## 330   DD    C    0
## 331    7    C    0
## 332  BBB    C    0
## 333   BB    C    0
## 334    B    C    0
## 335    C    C    0
## 336    0    C    0
## 337   DD    0    0
## 338    7    0    0
## 339  BBB    0    0
## 340   BB    0    0
## 341    B    0    0
```

```
## 342    C    0    0
## 343    0    0    0
```

```
# Calulating probabilities
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))
}

# make lookup table
prob <- c("DD" = 0.03, "7" = 0.03, "BBB" = 0.06,
  "BB" = 0.1, "B" = 0.25, "C" = 0.01, "0" = 0.52)

# lookup probabilities
combos$prob1 <- prob[combos$Var1]
combos$prob2 <- prob[combos$Var2]
combos$prob3 <- prob[combos$Var3]

head(combos, 3)
```

```
##   Var1 Var2 Var3 prob1 prob2 prob3
## 1   DD   DD   DD  0.03  0.03  0.03
## 2    7   DD   DD  0.03  0.03  0.03
## 3  BBB   DD   DD  0.06  0.03  0.03
```

```
# probablilities for each combination
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
```

```
# sum of probabilities
sum(combos$prob)
```

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

For Loops

```
# using for loop to calculate
combos$prize <- NA

head(combos, 3)
```

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

```r
#build a loop

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
```

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

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

```r
# Challenge: accounting for different score variations
score <- function(symbols) {

  diamonds <- sum(symbols == "DD")
  cherries <- sum(symbols == "C")

  # identify case
  # since diamonds are wild, only nondiamonds
  # matter for three of a kind and all bars
  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) {
    # diamonds count as cherries
    # so long as there is one real cherry
    prize <- c(0, 2, 5)[cherries + diamonds + 1]
  } else {
    prize <- 0
  }

  # double for each diamond
  prize * 2^diamonds
}
```

```r
# Calculating epected values
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] 0.934356
```

Chapter 11: Speed

Vectorizing Code:

```r
# Vectorize this code
change_symbols <- function(vec){
  for (i in 1:length(vec)){
    if (vec[i] == "DD") {
      vec[i] <- "joker"
    } else if (vec[i] == "C") {
      vec[i] <- "ace"
    } else if (vec[i] == "7") {
      vec[i] <- "king"
    }else if (vec[i] == "B") {
      vec[i] <- "queen"
    } else if (vec[i] == "BB") {
      vec[i] <- "jack"
    } else if (vec[i] == "BBB") {
      vec[i] <- "ten"
    } else {
      vec[i] <- "nine"
    }
  }
  vec
}

vec <- c("DD", "C", "7", "B", "BB", "BBB", "0")

change_symbols(vec)
```

```
## [1] "joker" "ace"   "king"  "queen" "jack"  "ten"   "nine"
```

```r
many <- rep(vec, 1000000)

system.time(change_symbols(many))
```

```
##    user  system elapsed
##    3.32    0.01    7.95
```

```r
# Solution:
vec[vec == "DD"]
```

```
## [1] "DD"
```

```r
vec[vec == "C"]
```

```
## [1] "C"
```

```r
vec[vec == "7"]
```

```
## [1] "7"
```

```r
vec[vec == "B"]
```

```
## [1] "B"
```

```r
vec[vec == "BB"]
```

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

```r
vec[vec == "BBB"]
```

```
## [1] "BBB"
```

```r
vec[vec == "0"]
```

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

```r
# Code that can shange each symbol for each case:
vec[vec == "DD"] <- "joker"
vec[vec == "C"] <- "ace"
vec[vec == "7"] <- "king"
vec[vec == "B"] <- "queen"
vec[vec == "BB"] <- "jack"
vec[vec == "BBB"] <- "ten"
vec[vec == "0"] <- "nine"

# combine into funciton and it runs 14 times facters
change_vec <- function (vec) {
  vec[vec == "DD"] <- "joker"
  vec[vec == "C"] <- "ace"
  vec[vec == "7"] <- "king"
  vec[vec == "B"] <- "queen"
  vec[vec == "BB"] <- "jack"
  vec[vec == "BBB"] <- "ten"
  vec[vec == "0"] <- "nine"

  vec
}

system.time(change_vec(many))
```

```
##    user  system elapsed
##    0.20    0.00    0.36
```

```
# even btter use lookup table
change_vec2 <- function(vec){
  tb <- c("DD" = "joker", "C" = "ace", "7" = "king", "B" = "queen",
    "BB" = "jack", "BBB" = "ten", "0" = "nine")
  unname(tb[vec])
}

system.time(change_vec(many))
```

```
##    user  system elapsed
##    0.25    0.02    0.33
```

Vectoruzed code in practice

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

mean(winnings)
```

```
## [1] 0.928232
```

```
# run fastre with vectorized code
# rewrit functios for this
get_many_symbols <- function(n) {
  wheel <- c("DD", "7", "BBB", "BB", "B", "C", "0")
  vec <- sample(wheel, size = 3 * n, replace = TRUE,
    prob = c(0.03, 0.03, 0.06, 0.1, 0.25, 0.01, 0.52))
  matrix(vec, ncol = 3)
}

play_many <- function(n) {
  symb_mat <- get_many_symbols(n = n)
  data.frame(w1 = symb_mat[,1], w2 = symb_mat[,2],
             w3 = symb_mat[,3], prize = score_many(symb_mat))
}


symbols <- matrix(
  c("DD", "DD", "DD",
    "C", "DD", "0",
    "B", "B", "B",
    "B", "BB", "BBB",
    "C", "C", "0",
    "7", "DD", "DD"), nrow = 6, byrow = TRUE)
```

```r
score_many <- function(symbols) {

  # Step 1: Assign base prize based on cherries and diamonds ---------
  ## Count the number of cherries and diamonds in each combination
  cherries <- rowSums(symbols == "C")
  diamonds <- rowSums(symbols == "DD")

  ## Wild diamonds count as cherries
  prize <- c(0, 2, 5)[cherries + diamonds + 1]

  ## ...but not if there are zero real cherries
  ### (cherries is coerced to FALSE where cherries == 0)
  prize[!cherries] <- 0

  # Step 2: Change prize for combinations that contain three of a kind
  same <- symbols[, 1] == symbols[, 2] &
    symbols[, 2] == symbols[, 3]
  payoffs <- c("DD" = 100, "7" = 80, "BBB" = 40,
    "BB" = 25, "B" = 10, "C" = 10, "0" = 0)
  prize[same] <- payoffs[symbols[same, 1]]

  # Step 3: Change prize for combinations that contain all bars ------
  bars <- symbols == "B" | symbols ==  "BB" | symbols == "BBB"
  all_bars <- bars[, 1] & bars[, 2] & bars[, 3] & !same
  prize[all_bars] <- 5

  # Step 4: Handle wilds ---------------------------------------------

  ## combos with two diamonds
  two_wilds <- diamonds == 2

  ### Identify the nonwild symbol
  one <- two_wilds & symbols[, 1] != symbols[, 2] &
    symbols[, 2] == symbols[, 3]
  two <- two_wilds & symbols[, 1] != symbols[, 2] &
    symbols[, 1] == symbols[, 3]
  three <- two_wilds & symbols[, 1] == symbols[, 2] &
    symbols[, 2] != symbols[, 3]

  ### Treat as three of a kind
  prize[one] <- payoffs[symbols[one, 1]]
  prize[two] <- payoffs[symbols[two, 2]]
  prize[three] <- payoffs[symbols[three, 3]]

  ## combos with one wild
  one_wild <- diamonds == 1

  ### Treat as all bars (if appropriate)
  wild_bars <- one_wild & (rowSums(bars) == 2)
  prize[wild_bars] <- 5

  ### Treat as three of a kind (if appropriate)
  one <- one_wild & symbols[, 1] == symbols[, 2]
```

```r
  two <- one_wild & symbols[, 2] == symbols[, 3]
  three <- one_wild & symbols[, 3] == symbols[, 1]
  prize[one] <- payoffs[symbols[one, 1]]
  prize[two] <- payoffs[symbols[two, 2]]
  prize[three] <- payoffs[symbols[three, 3]]

  # Step 5: Double prize for every diamond in combo -------------------
  unname(prize * 2^diamonds)

}

system.time(play_many(10000000))
```

```
##    user  system elapsed
##    3.40    1.17   10.61
```