Programs

20173204 곽명빈

2020-05-22

Intro

Payouts

```
wheel <- c("DD", "7", "BBB", "BB", "B", "C", "0")
Combination <- wheel %>%
  head(-1) %>%
  sapply(., rep, each = 3) %>%
  apply(., MARGIN = 2, paste, collapse = " ") %>%
  unname %>%
  c(., "Any combination of bars", "Double Cherries", "Single Cherry")
Prizes <- c(100, 80, 40, 25, 10, 10, 5, 5, 2)
data.frame(Combination, Prizes, stringsAsFactors = FALSE)
```

```
##               Combination Prizes
## 1                DD DD DD    100
## 2                   7 7 7     80
## 3             BBB BBB BBB     40
## 4                BB BB BB     25
## 5                   B B B     10
## 6                   C C C     10
## 7 Any combination of bars      5
## 8         Double Cherries      5
## 9           Single Cherry      2
```

```
score(c("DD", "DD", "DD"))
## 800
```

```
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"  "DD"
```

```
table(replicate(1000, get_symbols()))
```

```
##
##    0    7    B   BB  BBB    C   DD
## 1596   79  723  318  166   28   90
```

```
table(replicate(1000, get_symbols()))/3000
```

```
##
##          0          7          B         BB        BBB          C         DD
## 0.52866667 0.02433333 0.24100000 0.10666667 0.05766667 0.01166667 0.03000000
```

표준오차는 1%

```
round(table(replicate(1000, get_symbols()))/3000, digits = 2)
```

```
##
##    0    7    B   BB  BBB    C   DD
## 0.52 0.03 0.25 0.09 0.06 0.01 0.03
```

```
round(table(replicate(10000, get_symbols()))/30000, digits = 2)
```

```
##
##    0    7    B   BB  BBB    C   DD
## 0.52 0.03 0.25 0.10 0.06 0.01 0.03
```

Strategy

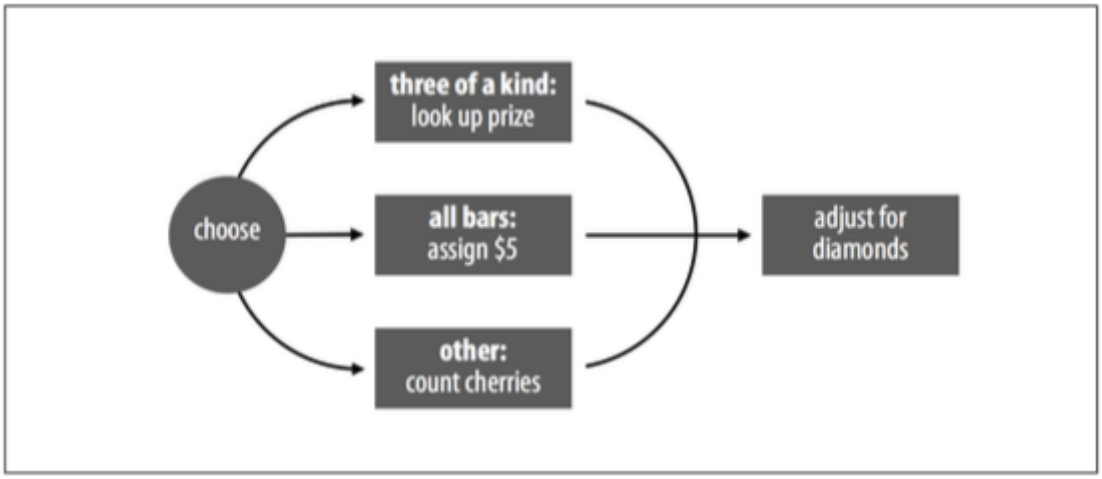
1. Break complex taks into simple subtasks
2. Use concrete examples
3. Describe your solutions in English(Korean), then convert them to `R`.

Sequential Steps

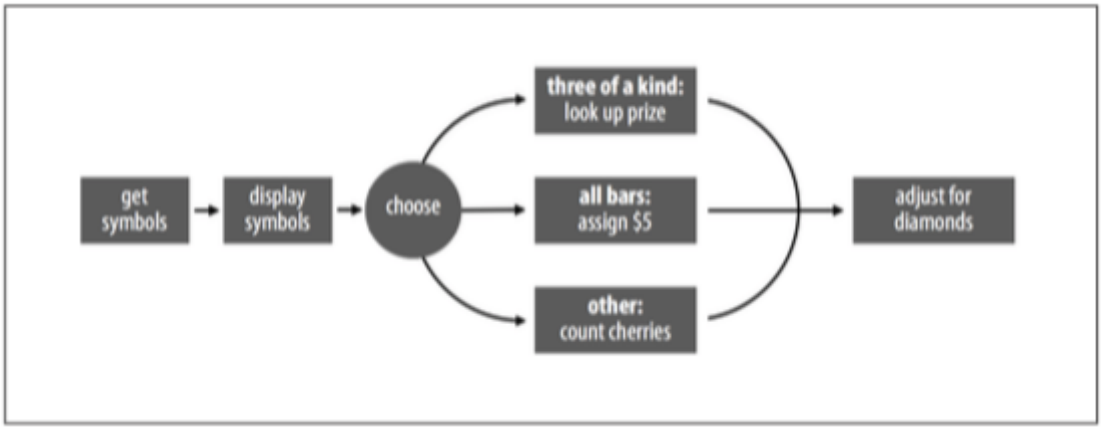
```
play <- function() {

#> Step 1 : generate symbols
symbols <- get_symbols()

#> Step 2 : display the symbols
print(symbols)

#> step 3 : score the symbols
score(symbols)
}
```

Parallel Cases

Score function structure



The complete slot machine simulation



if Statements

```
if (this) {
that
}
```

```
# num <- -2
num <- 4
if (num < 0) {
  num <- num * -1
}
num
```

```
## [1] 4
```

```
all(c(TRUE, FALSE))
```

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

```
any(c(TRUE, FALSE))
```

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

```
num <- -1
if (num < 0) {
  print("num is negative.")
  print("Don't worry, I'll fix it.")
  num <- num * -1
  print("Now num is positive.")
}
```

```
## [1] "num is negative."
## [1] "Don't worry, I'll fix it."
## [1] "Now num is positive."
```

```
num
```

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

Quiz A

```
x <- -1
if (3 == 3){
  x <- 2
}
x
```

```
## [1] 2
```

Quiz B

```
x <- 1
if (TRUE) {
  x <- 2
}
x
```

```
## [1] 2
```

Quiz C

```
x <- 1
if (x == 1) {
  x <- 2
  if (x == 1) {
    x <- 3
  }
}
x
```

```
## [1] 2
```

else Statements

```
if (this) {
  Plan A
} else {
  Pla B
}
```

```
pi
```

```
## [1] 3.141593
```

```
a <- pi
dec <- a - trunc(pi)
dec
```

```
## [1] 0.1415927
```

```
if(dec >= 0.5) {
  a <- trunc(a) + 1
} else {
  a <- trunc(a)
}
a
```

```
## [1] 3
```

```
a <- 2
b <- 2
if (a > b) {
  print("A wins!")
} else if (a < b) {
  print("B wins!")
} else {
  print("Tie.")
}
```

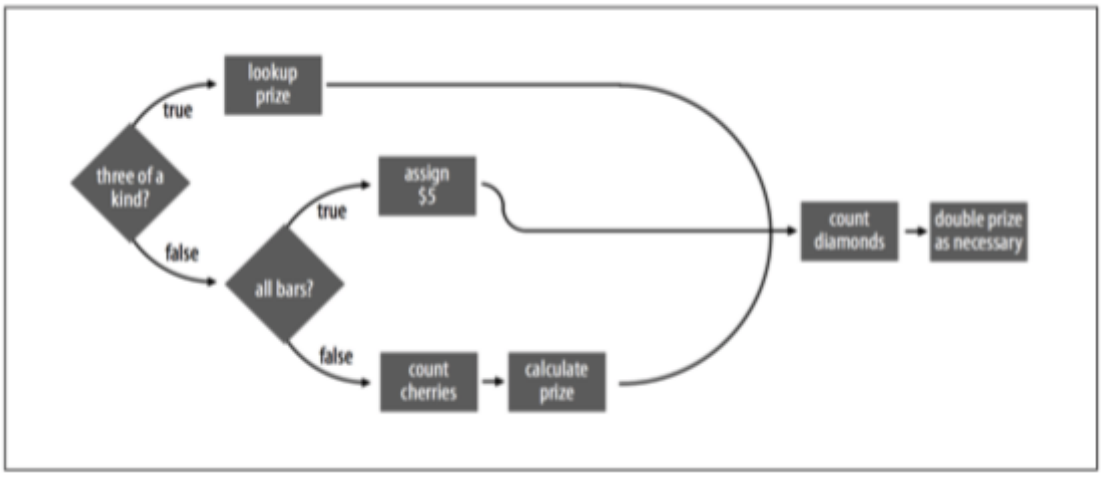
```
## [1] "Tie."
```

Test whether the symbols are three of a kind

Code skeleton

```
if ( #> Case 1: all the same) {
  prize <- #> look up the prize
} else if ( #> Case 2: all bars) {
  prize <- #> assign $5
} else {
  #> Count cherries
  prize <- calculate a prize
}
#> count diamonds
#> double the prize is necessary
```

Score with `if`, `else`




```
score <- function(symbols){

#> calculate a prize

  prize
}
```

Three of the same symbols

```
symbols <- c("7", "7", "7")
symbols[1] == symbols[2] & symbols[2] == symbols[3]
```

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

```
symbols == symbols[1]
```

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

```
all(symbols == symbols[1])
```

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

```
unique(symbols)
```

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

```
length(unique(symbols))
```

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

```
length(unique(symbols)) == 1
```

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

Code update

```
same <- symbols[1] == symbols[2] && symbols[2] == symbols[3]

if (same) {
  prize <- #> look up the prize
} else if ( #> Case 2: all bars) {
  prize <- #> assign $5
} else {
  #> count cherries
  prize <- #> calculate a prize
}

#> count diamonds
#> double the prize if necessary
```

All the symbols are a type of bars

```
#> Three of the same symbols는 아닌 경우
symbols <- c("B", "BBB", "BB")
symbols %in% c("B", "BB", "BBB")
```

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

```
symbols <- c("B", "BBB", "B")
symbols %in% c("B", "BB", "BBB")
```

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

```
match(symbols, c("B", "BB", "BBB"))
```

```
## [1] 1 3 1
```

```
all(symbols %in% c("B", "BB", "BBB"))
```

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

```
same <- length(unique(symbols)) == 1
bars <- symbols %in% c("B", "BB", "BBB")
same
```

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

```
all(bars)
```

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

```
#> 셋 다 같은 bar 인 경우
symbols <- rep("B", 3)
same <- length(unique(symbols)) == 1
same
```

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

```
bars <- symbols %in% c("B", "BB", "BBB")
all(bars)
```

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

Code update

```
same <- symbols[1] == symbols[2] && symbols[2] == symbols[3]
bars <- symbols %in% c("B", "BB", "BBB")

if (same) {
  prize <- #> look up the prize
} else if (all(bars)) {
  prize <- #> assign $5
} else {
  #> count cherries
  prize <- #> calculate a prize
}

#> count diamonds
#> double the prize if necessary
```

Lookup Tables

Read pp. 131 for complicated `if` statements for comparison.

```
payouts <- c("DD" = 100, "7" = 80, "BBB" = 40, "BB" = 25, "B" = 10, "C" = 10, "0" = 0)
payouts
```

```
##  DD   7 BBB  BB   B   C   0
## 100  80  40  25  10  10   0
```

```
payouts["DD"]
```

```
##  DD
## 100
```

```
payouts["B"]
```

```
##  B
## 10
```

```
unname(payouts["DD"])
```

```
## [1] 100
```

```
symbols <- c("7", "7", "7")
symbols[1]
```

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

```
payouts[symbols[1]]
```

```
##  7
## 80
```

```
prize <- unname(payouts[symbols[1]])
prize
```

```
## [1] 80
```

```
symbols <- c("C", "C", "C")
payouts[symbols[1]]
```

```
##  C
## 10
```

```
prize <- unname(payouts[symbols[1]])
prize
```

```
## [1] 10
```

Code update (Case 2 included)

```
same <- length(unique(symbols)) == 1
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 {
  # count cherries
  prize <- #> calculate a prize
}
#> count diamonds
#> double the prize if necessary
```

Count Cherries and Diamonds

```
symbols <- c("C", "DD", "C")
symbols == "C"
```

```
## [1]  TRUE FALSE  TRUE
```

```
sum(symbols == "C")
```

```
## [1] 2
```

```
cherries <- sum(symbols == "C")
cherries
```

```
## [1] 2
```

```
cherries + 1
```

```
## [1] 3
```

```
sum(symbols == "DD")
```

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

```
diamonds <- sum(symbols == "DD")
2 ^ diamonds
```

```
## [1] 2
```

```
if (cherries == 2) {
  prize <- 5
} else if (cherries == 1) {
  prize <- 2
} else {}
  prize <- 0
}
```

Application of Lookup table

```
#> Cherry가 두개인 경우
symbols <- c("C", "DD", "C")
symbols == "C"
```

```
## [1]  TRUE FALSE  TRUE
```

```
sum(symbols == "C")
```

```
## [1] 2
```

```
cherries <- sum(symbols == "C")
cherries
```

```
## [1] 2
```

```
cherries + 1
```

```
## [1] 3
```

```
c(0, 2, 5)[cherries + 1]
```

```
## [1] 5
```

```
#> Cherry가 하나인 경우
symbols <- c("C", "DD", "B")
symbols == "C"
```

```
## [1]  TRUE FALSE FALSE
```

```
sum(symbols == "C")
```

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

```
cherries <- sum(symbols == "C")
cherries
```

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

```
cherries + 1
```

```
## [1] 2
```

```
c(0, 2, 5)[cherries + 1]
```

```
## [1] 2
```

```
#> Cherry가 없는 경우
symbols <- c("DD", "DD", "B")
symbols == "C"
```

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

```
sum(symbols == "C")
```

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

```
cherries <- sum(symbols == "C")
cherries
```

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

```
cherries + 1
```

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

```
c(0, 2, 5)[cherries + 1]
```

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

```
#> DD 갯수 세어보기
sum(symbols == "DD")
```

```
## [1] 2
```

```
diamonds <- sum(symbols == "DD")
2 ^ diamonds
```

```
## [1] 4
```

Code update

```
same <- length(unique(symbols)) == 1
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 <- prize * 2 ^ diamonds
prize
```

Final Version

```
score <- function(symbols) {
  same <- length(unique(symbols)) == 1
  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
}
```

Code Comments

```
score <- function(symbols) {
  #> identify cases
  same <- length(unique(symbols)) == 1
  bars <- symbols %in% c("B", "BB", "BBB")

  #> get prize
  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]
  }

  #> adjust for diamonds
  diamonds <- sum(symbols == "DD")
  prize * 2 ^ diamonds
}
```

How to play

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

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

```
## [1] 5
```

```
play()
```

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

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

```
play()
```

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

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

```
play()
```

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

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

```
replicate(10, play())
```

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

```
##  [1] 0 0 0 0 0 0 5 0 0 0
```

```
play()
```

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

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

```
one_play <- play()
```

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

```
one_play
```

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

Save

```
save.image(file = "./Programs.RData")
```

Comments

이번시간에는 실제로 작동하는 슬롯머신을 만드는법을 배울수있는 시간이였습니다. wheel에 문자를 저장하고 각 문자의 확률을 저장하는 방법을 알게 되었습니다. 또한 IF를 통해 조건문을 만들수 있게되었고, match를 통해 symbol이 어느위치에 나오는지에 대한 방법을 알수있는 계기가 되었습니다. 이번 수업을 통해 팀프로젝트에 하려던 추출을 할 수있게 된 수업이여서 의미있는 강의였습니다.