Deck of Cards

Silas Tittes

July 2, 2015

Yesterday I was day dreaming about probability and decks of cards. Gambling and probability are like old high school pals. One went on to college and got a nice job in science. She plays tennis on the weekends and gives lectures at fine ivy league institutions (among others). The other is a wealthy alcoholic that spends his time in Vegas casinos (among others). They still get together on a rare occasion, but it doesn't usually end well for anyone (card counters). Anyways, I realized it how easy it would be to simulate a deck of cards using (what else?) R.

Here is the deck of cards:

```
nums <- as.character(2:10)  #numbered cards
types <- c("Ace", "Jack", "Queen",
    "King", nums)  #face cards and numbered cards (the ranks)
suites <- c("Hearts", "Spades",
    "Diamonds", "Clubs")  #the suites

# repeat each value in the card
# types vector by the number of
# suites, and repeat each value
# in the suites vector by the
# number of types. Then paste
# the two output vectors
# together and seperate the
# items by '_of_'
deck <- paste(rep(types, length(suites)),
    rep(suites, length(types)),
    sep = "_of_")
```

Did I do it right?

```
52 == length(deck)  #hopefully 52 (There's only room for one joker in this blog post.)
```

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

```
52 == length(unique(deck))  #each card is like a snowflake
```

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

```
# split the string back into
# the suites and types
broken_deck <- unlist(strsplit(deck,
    split = "_of_"))

# the %in% operator is the
# bomb!
rep(13, 4) %in% table(broken_deck[1:length(broken_deck)%%2/3 ==
    0])  #count of each suite
```

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

```
rep(4, 13) %in% table(broken_deck[1:length(broken_deck)%%2/3 !=
    0])  #count of each rank
```

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

```
print(deck, quote = F)  #remove the quotes. quotes are for gamblers.
```

```
##  [1] Ace_of_Hearts     Jack_of_Spades    Queen_of_Diamonds
##  [4] King_of_Clubs     2_of_Hearts       3_of_Spades
##  [7] 4_of_Diamonds     5_of_Clubs        6_of_Hearts
## [10] 7_of_Spades       8_of_Diamonds     9_of_Clubs
## [13] 10_of_Hearts      Ace_of_Spades     Jack_of_Diamonds
## [16] Queen_of_Clubs    King_of_Hearts    2_of_Spades
## [19] 3_of_Diamonds     4_of_Clubs        5_of_Hearts
## [22] 6_of_Spades       7_of_Diamonds     8_of_Clubs
## [25] 9_of_Hearts       10_of_Spades      Ace_of_Diamonds
## [28] Jack_of_Clubs     Queen_of_Hearts   King_of_Spades
## [31] 2_of_Diamonds     3_of_Clubs        4_of_Hearts
## [34] 5_of_Spades       6_of_Diamonds     7_of_Clubs
## [37] 8_of_Hearts       9_of_Spades       10_of_Diamonds
## [40] Ace_of_Clubs      Jack_of_Hearts    Queen_of_Spades
## [43] King_of_Diamonds  2_of_Clubs        3_of_Hearts
## [46] 4_of_Spades       5_of_Diamonds     6_of_Clubs
## [49] 7_of_Hearts       8_of_Spades       9_of_Diamonds
## [52] 10_of_Clubs
```

I love R! 4 simple lines and you have a deck of cards to play with. Now with 4 more lines, I'll show you how to write your own solitaire program. No, not really, but maybe one day, with like 4,000 lines at least. However, I will make a few simple card sampling functions for funsies.

```
# function with no input
shuffle <- function() {
    sample(deck, size = length(deck),
        replace = F)
}
# same function with input
shuffle <- function(n.cards) {
    sample(deck, size = n.cards,
        replace = F)
}
# function with no input
draw <- function() {
    sample(deck, size = length(deck),
        replace = T)  #notice TRUE not FALSE
}
# same function, different
# inputs (nothing versus number
# of samples)
draw <- function(n.cards) {
    sample(deck, size = n.cards,
        replace = T)  #notice TRUE not FALSE
}
```

Here I made two functions, shuffle and draw. Each function is actually two functions with the same name. I could have made all this into one function, but shuffle and draw a different enough conceptually, so I thought I would keep them separate for demonstration purposes. When the shuffle function receives no input, it randomly grabs from the original deck WITHOUT replacement 52 times. Since it grabs randomly, it is in a sense shuffling the card deck. The shuffle function also takes an integer as an input, in which case n number of cards are drawn at random, again without replacement. Draw is the same as shuffle, but it samples WITH replacement. Meaning the same card could be sampled more than once in a series of draws. The two functions really only differ by the letter T and F (for true and false) in the sample function. I should mention this is a simple form of function overloading, where giving input or no input changes the function behavior on the user's end. Since R doesn't make you declare variable types, function overloading with alternative types of variables as inputs doesn't work the same way function overloading works in something like c++. As far as I can tell R instead uses what are called methods (see ?methods and ?setMethods). I won't go into setting methods here (frankly I haven't really tried setting methods myself). Instead we will stick to this simple hack-esh version of overloading that is probably useless in realistic applications. Okay, lets shuffle and draw the deck shall we?

```
# first use of overloaded
# function returns all the
shuffle()
```

```
##  [1] "Queen_of_Hearts"   "Jack_of_Spades"    "7_of_Diamonds"
##  [4] "2_of_Hearts"       "2_of_Clubs"        "Ace_of_Hearts"
##  [7] "Jack_of_Clubs"     "6_of_Clubs"        "8_of_Diamonds"
## [10] "Ace_of_Spades"     "3_of_Diamonds"     "3_of_Spades"
## [13] "Jack_of_Diamonds"  "King_of_Hearts"    "10_of_Clubs"
## [16] "5_of_Diamonds"     "9_of_Diamonds"     "9_of_Clubs"
## [19] "King_of_Diamonds"  "6_of_Hearts"       "8_of_Clubs"
## [22] "Ace_of_Clubs"      "7_of_Spades"       "King_of_Spades"
## [25] "3_of_Clubs"        "5_of_Spades"       "5_of_Clubs"
## [28] "4_of_Spades"       "9_of_Spades"       "8_of_Hearts"
## [31] "10_of_Diamonds"    "10_of_Spades"      "Queen_of_Spades"
## [34] "King_of_Clubs"     "10_of_Hearts"      "7_of_Clubs"
## [37] "7_of_Hearts"       "8_of_Spades"       "Queen_of_Clubs"
## [40] "Ace_of_Diamonds"   "9_of_Hearts"       "2_of_Diamonds"
## [43] "4_of_Hearts"       "4_of_Clubs"        "Jack_of_Hearts"
## [46] "Queen_of_Diamonds" "3_of_Hearts"       "6_of_Spades"
## [49] "4_of_Diamonds"     "2_of_Spades"       "6_of_Diamonds"
## [52] "5_of_Hearts"
```

```
shuffle(2)
```

```
## [1] "5_of_Diamonds" "4_of_Clubs"
```

```
draw()
```

```
##  [1] "Queen_of_Hearts"   "8_of_Spades"       "Queen_of_Diamonds"
##  [4] "King_of_Hearts"    "9_of_Clubs"        "King_of_Hearts"
##  [7] "8_of_Clubs"        "9_of_Spades"       "4_of_Diamonds"
## [10] "6_of_Diamonds"     "3_of_Diamonds"     "3_of_Clubs"
## [13] "4_of_Spades"       "8_of_Clubs"        "6_of_Clubs"
## [16] "2_of_Hearts"       "9_of_Spades"       "6_of_Clubs"
## [19] "King_of_Diamonds"  "9_of_Clubs"        "9_of_Diamonds"
## [22] "3_of_Spades"       "King_of_Clubs"     "7_of_Spades"
## [25] "9_of_Hearts"       "3_of_Clubs"        "8_of_Spades"
## [28] "Jack_of_Diamonds"  "King_of_Clubs"     "Queen_of_Diamonds"
## [31] "King_of_Diamonds"  "3_of_Spades"       "3_of_Spades"
## [34] "4_of_Clubs"        "King_of_Diamonds"  "King_of_Spades"
## [37] "5_of_Clubs"        "5_of_Hearts"       "3_of_Clubs"
## [40] "6_of_Clubs"        "King_of_Hearts"    "6_of_Spades"
## [43] "Ace_of_Clubs"      "Ace_of_Spades"     "Queen_of_Hearts"
## [46] "Queen_of_Diamonds" "9_of_Diamonds"     "King_of_Spades"
## [49] "5_of_Clubs"        "5_of_Diamonds"     "10_of_Clubs"
## [52] "Jack_of_Clubs"
```

```r
draw(2)
```

```
## [1] "3_of_Diamonds"   "Ace_of_Diamonds"
```

```r
# lets see how often we get a
# particular card on a large
# number of draws and see how
# close to 1/52 we get
card <- "Queen_of_Spades"
n.draw <- 1e+05
exp <- round(1/52 * n.draw, 0)
names(exp) <- "counts"
bunch_o_draws <- draw(n.draw)
obs <- sum(card == bunch_o_draws)
print(cbind(obs, exp))
```

```
##         obs  exp
## counts 1858 1923
```

Overall, very silly. Hopefully though it demonstrates the ease in which fairly complex tasks can be accomplished very easily in R. I use the sample function daily. Perhaps in a later post I could use the deck of cards for a better application.

For now I'll leave it at that. Next time, additive alleles and drift!