HW8 Combination lock

author: Jamie Ash
date: 2022-10-11

Calculating it with code

I chose to do the combination lock question. I wrote out some code that calculates all possible combinations of a lock with a any number of buttons and two entries. I have not generalized it for a lock with any number of entries. I did my best writing the function out as an equation.

```
# Total outcomes fo draws (k) out of n elements
binom = function(n, k){
  factorial(n) / (factorial(k) * factorial(n-k))
}

# total combinations for one press with a combo lock with a number of buttons
lock = function(buttons = 5){
  sum(binom(k = seq(1, buttons), n = buttons))
  }

# total combinations for two presses with a combo lock with a number of buttons
combo_lock = function(buttons = 5) {
  psw = vector()
  for(press in 1:(buttons-1)) {
    psw[press] = binom(k = press, n = buttons) * lock(buttons-press)
    }
  sum(psw)
}
```

The `binom()` function is vectorized, so if I enter a vector of [1,2,3,4,5] it will return a vector for five k values 1 to 5. The `lock()` function calculates all the possible combinations for a lock with only one entry and any number of buttons. That is the sum of the binomial function from 1 to the number of buttons on the lock. Lastly, the `combo_lock()` function calculates the total number of combinations for a lock with any number of buttons and two entries.

```
# combo lock with five buttons and 2 entries
buttons = 5
pass = combo_lock(buttons)
pass
```

```
## [1] 180
```

So a combination lock with 5 buttons can 180 passwords.

Translating code to equation

I represent this function as an equation bellow. I've never been good with the Σ symbol so I may have made a mistake translating the code to an equation.

$$\sum_{k=1}^{n-1}\left[\binom{k}{n}\times\sum_{x=1}^{n-k}\binom{x}{n-k}\right] \tag{1}$$

I assume that the length of the code set is 2. This means there is no $\binom{0}{5}$ for the first or second entry in the code set. This translates to $n-1$ at the top of the leftmost Σ, meaning the first entry in the code does not have 5 digits (leaving 0 buttons left for the second entry).

Because the combination lock only has five buttons and two entries it's possible to work through the function without too much trouble.

$$\sum_{k=1}^{n-1}\left[\binom{k}{n}\times\sum_{x=1}^{n-k}\binom{x}{n-k}\right] \tag{1}$$

$$\begin{aligned}
\binom{1}{5}\times\left[\binom{4}{4}+\binom{3}{4}+\binom{2}{4}+\binom{1}{4}\right] &= 75\\
\binom{2}{5}\times\left[\binom{3}{3}+\binom{2}{3}+\binom{1}{3}\right] &= 70\\
\binom{3}{5}\times\left[\binom{2}{2}+\binom{1}{2}\right] &= 30\\
\binom{4}{5}\times\left[\binom{1}{1}\right] &= 5
\end{aligned}$$

$$75+70+30+5=180$$

Where n is the number of buttons, k is the number of buttons pressed for each combination. The total combinations for the second number in the password set can be calculated as the number of possible combinations for a lock with one entry and a number of buttons equal to the total number of buttons minus the number of buttons pressed in the first.

Extending to n presses

I was able to generalize a lock with n buttons with only two presses but not for up to n presses. I'm just going to calculate it the long way.

```
# press one button
press_1 = lock(5)

# press 2 buttons
press_2 = combo_lock(5)

# press 3 buttons
press_3 = sum(
  binom(k = 1, n = buttons) * binom(k = 1, n = 4) * lock(3),
  binom(k = 1, n = buttons) * binom(k = 2, n = 4) * lock(2),
  binom(k = 1, n = buttons) * binom(k = 3, n = 4) * lock(1),
  binom(k = 2, n = buttons) * binom(k = 1, n = 3) * lock(3),
  binom(k = 2, n = buttons) * binom(k = 2, n = 3) * lock(2),
  binom(k = 3, n = buttons) * binom(k = 1, n = 2) * lock(1)
  )

# press 4 buttons
press_4 = sum(
  binom(k = 1, n = buttons) * binom(k = 1, n = 4) * binom(k = 1, n = 3) * lock(2),
  binom(k = 1, n = buttons) * binom(k = 1, n = 4) * binom(k = 2, n = 3) * lock(1),
  binom(k = 1, n = buttons) * binom(k = 2, n = 4) * binom(k = 1, n = 2) * lock(1),
  binom(k = 2, n = buttons) * binom(k = 1, n = 3) * binom(k = 1, n = 2) * lock(1)
  )

# press 5 buttons
press_5 = factorial(5)
```

The total pass codes will be the sum of 1 press, 2 press, 3 press, 4 press and 5 press codes.

```
press_1 + press_2 + press_3 + press_4 + press_5
```

```
## [1] 1261
```

So that's what I have. Still working on a generalized equation or writing a function that can calculate the total number of pass codes for a lock with any number of buttons.