Final exam review

Benchmarking: goal is to compare performance of different methods / functions / procedures

(both time & memory)

bench::mark (....)

↳ returns time it takes to run, # iterations / second, memory used

(want faster performance, less memory)

Q 30 (Final exam practice):

40 cards, 4 different colors

Each color has cards 1 - 10

Color:	1	2	3	4
Cards: #s	1, 2, ..., 10	1, 2, ..., 10	1, 2, ..., 10	1, 2, ..., 10

Choose two of the 40 cards at random (w/o replacement). Want the probability that the two cards have different #s and different colors

Probability simulation to estimate probability:

- set up cards
  - represent color of each card

```
colors <- c(rep("red", 10), rep("blue", 10), rep("green", 10),
            rep("yellow", 10))
```

  - represent # of each card

```
numbers <- rep(1:10, 4)
```

← 1, 2, ..., 10, 1, 2, ..., 10, ...

- choose 2 cards at random

```
cards <- sample(1:40, 2, replace=F)
```

- compare #s & compare colors

```
colors[cards[1]] != colors[cards[2]]
numbers[cards[1]] != numbers[cards[2]]
```

- store result (e.g., start w/ result = 0, add 1 when cards have different #s & diff. colors)
- repeat many times (for loop!)

```
nsim <- 100

result <- 0
colors <- c(rep("red", 10), rep("blue", 10), rep("green", 10),
                                              rep("yellow", 10))

numbers <- rep(1:10 , 4)
for (i in 1:nsim) {

    cards <- sample(1:40 , 2, replace=F)
    if ((colors[cards[1]] != colors[cards[2]]) &
        (numbers[cards[1]] != numbers[cards[2]])) {
            result <- result +1
    }
}
result/nsim
```

Some helpful regular expression pieces:

\\d : a digit

+ : "appears one or more times"

(e.g. \\d+ a sequence of digits)

. : "any character"

(.+ any sequence of characters)

[ ] : character class

(e.g., [abc] : match an a, b, or c)

[abc]+ could match aabaabc
abba
abbcccb

[^] : everything except specified characters

[^abc] everything except a, b, or c

^ : anchor for beginning of a string

$: anchor for end of a string

(?=) : (positive) lookahead

(?<=) : (positive) lookbehind

\\. : actually a period

\\(: actually a left parenthesis
(etc.)
\\ escapes the special characters

Eg. want a pattern between , and ;
(?<=,).+(?=;)