# Excercise - Optimal Stopping

**Core Empirical Research Methods - Summer Work**

Toby Wang

2023-08-04

## Q1

### First attempt

The basic thought process behind this version of the simulation setup is that:

- Randomly sample the ranks of the phase 1 supervisors
- Order the list of supervisors from best score to worst score, and select the one from the first value of that vector, as well as his rank (`test_list_best`)
- Then sample the supervisors in phase 2 by randomly assiging each supervior with a rank, exclusing the ranks that have been given to the phase 1 supervisors (`super_list_rank`)
- Finally, cycle through each supervior, checking if their rank is higher than the largest out of all professors in phase 1, returning the rank of the processor I end up with.

```
set.seed(69)
k <- 40
n <- 50
supervisor_sims <- \(n, k) {
  test_list <- c(1:k)
  test_list_rank <- sample(1:n, k, replace = FALSE)
  test_list_ordered <- test_list[order(test_list_rank, decreasing = TRUE)]
  test_list_best <- c(test_list_ordered[1], max(test_list_rank))
  super_list <- c(k+1, n)
  super_list_rank <- sample(1:n, n, replace = FALSE)
  super <- which(!super_list_rank %in% test_list_rank)
  super_list_rank <- super_list_rank[super]
  i <- 1
  while (super_list_rank[i] > min(test_list_rank) & i < n-k) {
    i <- i+1
  }
  super_list_rank[i]
```

```
}
set.seed(69)
mean(map_dbl(1:100000, \(i) supervisor_sims(50, 40)))
```

```
[1] 20.95339
```

### Second Attempt

Even though the result on the first is correct (as it matched with the code below when sampling them many times), I found a simplier way to produce the same output. The basic idea is that we sample all the superviors rank in one go (this is probably the easiest way to do it).

```
set.seed(69)
supervisor_sim <- \(n, k) {
  # Randomly generate scores for n supervisors
  supervisors_rank <- sample(1:n, n, replace = FALSE)

  # Sample k supervisors for phase 1
  phase_one_rank <- supervisors_rank[1:k]
  min_rank <- min(phase_one_rank)

  # Start phase 2
  for (i in (k+1):n) {
    if (supervisors_rank[i] < min_rank) {
      return(supervisors_rank[i])
    }
  }

  # If no supervisor found in phase 2, return last one
  return(supervisors_rank[n])
}

mean(map_dbl(1:100000, \(i) supervisor_sim(50, 40)))
```

```
[1] 21.02788
```

## Q2

To do this we just use the map function to sample lots of times and then calculate what percentage of those is ranked at number one

```
set.seed(6969)
get_prob_preferred <- \(n, k, sample_size = 10000) {
  samples <- map_dbl(1:sample_size, \(i) supervisor_sim(n, k))
  sum(samples == 1) / sample_size
}
get_prob_preferred(50, 5)
```

```
[1] 0.2422
```

Just to check that the first simulation works as well:

```
set.seed(6969)
get_prob_preferredd <- \(n, k, sample_size = 10000) {
  samples <- map_dbl(1:sample_size, \(i) supervisor_sims(n, k))
  sum(samples == 1) / sample_size
}
get_prob_preferredd(50, 5)
```

```
[1] 0.2399
```

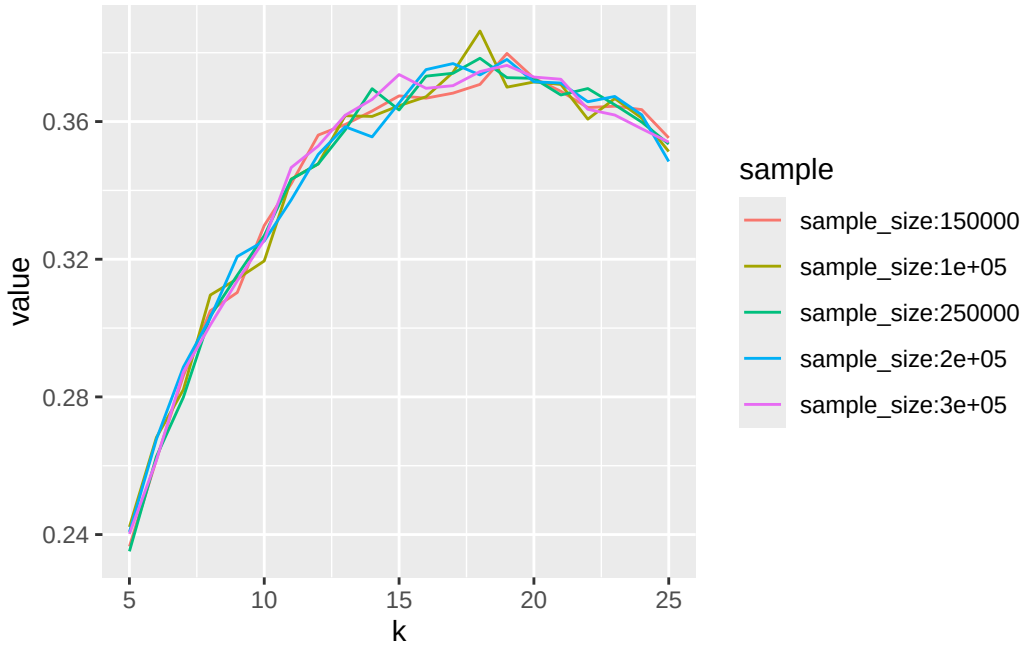
## Q3

```
set.seed(6969)
k_50 <- c(5:25)

one_map <- \(k_50, sample_size){
   map_dbl(k_50, \(k_50) get_prob_preferred(50, k_50, sample_size))
}
n <- 5
sample_seq <- seq(from = 1e4, by = 5e3, length.out = n)
names(sample_seq) <- paste0("sample_size:", seq(from = 1e5, by = 5e4, length.out = n))

df <- map_dfc(sample_seq, \(sample_seq) one_map(k_50, sample_seq)) |>
  mutate(k = c(5:25)) |>
  pivot_longer(-k, names_to = "sample", values_to = "value")

df |>
  ggplot(aes(x = k, y = value, color = sample)) +
  geom_line()
```

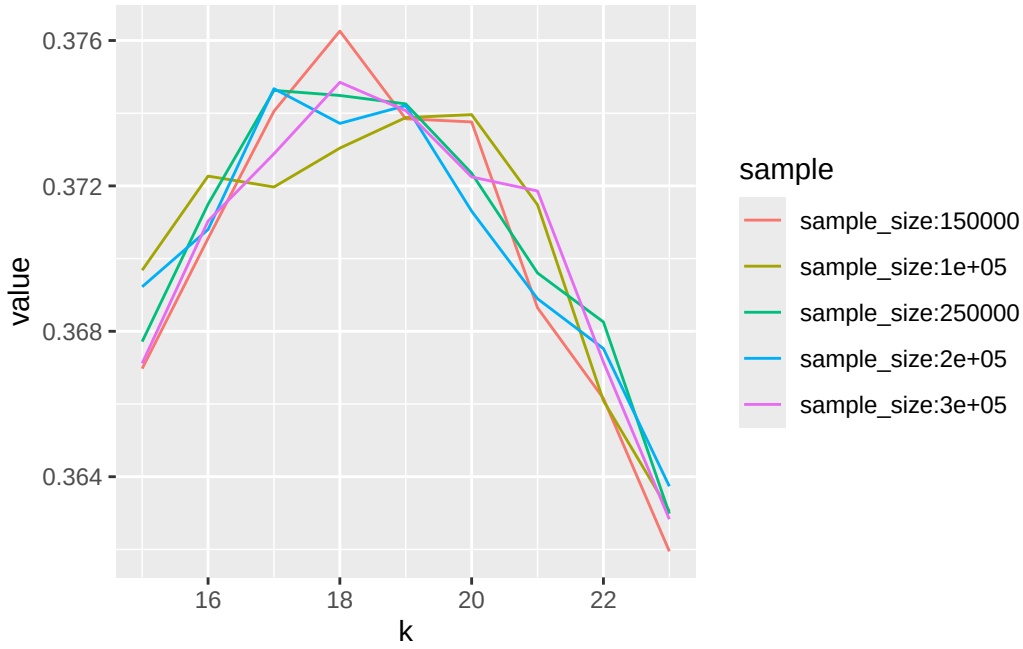



Okay... Let's now look at the 15-23 range...

```
set.seed(6969)
k_50 <- c(15:23)

one_map <- \(k_50, sample_size){
   map_dbl(k_50, \(k_50) get_prob_preferred(50, k_50, sample_size))
}
n <- 5
sample_seq <- seq(from = 1e5, by = 5e4, length.out = n)
names(sample_seq) <- paste0("sample_size:", seq(from = 1e5, by = 5e4, length.out = n))

map_dfc(sample_seq, \(sample_seq) one_map(k_50, sample_seq)) |>
  mutate(k = c(15:23)) |>
  pivot_longer(-k, names_to = "sample", values_to = "value") |>
  ggplot(aes(x = k, y = value, color = sample)) +
  geom_line()
```

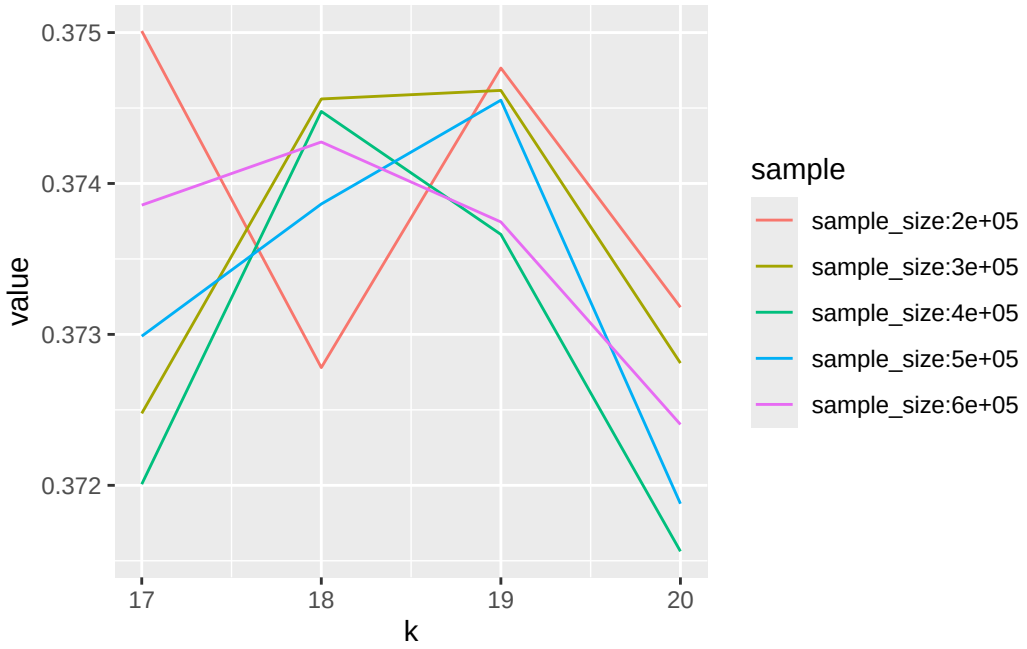



Now let's narrow this down further, say 16-21

```
set.seed(6969)
k_50 <- c(17:20)

one_map <- \(k_50, sample_size){
   map_dbl(k_50, \(k_50) get_prob_preferred(50, k_50, sample_size))
}
n <- 5
sample_seq <- seq(from = 2e5, by = 1e5, length.out = n)
names(sample_seq) <- paste0("sample_size:", seq(from = 2e5, by = 1e5, length.out = n))

map_dfc(sample_seq, \(sample_seq) one_map(k_50, sample_seq)) |>
  mutate(k = c(17:20)) |>
  pivot_longer(-k, names_to = "sample", values_to = "value") |>
  ggplot(aes(x = k, y = value, color = sample)) +
  geom_line()
```




It's hard to say for sure, but it seems as if the optimal value (the number of supervisors when `n=50` at phase one) is either at 18 or 19. I'm not sure if there's a value `k` that the simulation would converge on and in any case it took really long to run.