Exercise - Optimal Stopping

In this problem you will use simulation explore a simple and slightly silly example of an *optimal stopping* problem. Such problems crop up throughout economics, probability and statistics, and operations research. For an analytical approach to the problem outlined below see Fifty Challenging Problems in Probability with Solutions by Frederick Mosteller. For an approachable overview of optimal stopping problems, see Brian Christian and Tom Griffiths' recent book Algorithms to Live By. The problem that I describe below is sometimes called the "secretary problem" because of the story that typically goes along with it. I've written a different story, so mine is better called the "MPhil supervisor problem."

Soon you will have to choose your MPhil thesis supervisor. If you could meet and talk with all possible supervisors, you'd be able to rank them from best fit to worst fit, depending on your research interests. (For simplicity assume that there are no ties.) Somewhere in the department is your *most preferred* supervisor: you just have to find them!

If you meet with a potential supervisor, you will immediately learn the quality of fit with your research interests. You can think of this as a "supervisor score" that is revealed during the meeting. At the end of the meeting your potential supervisor will make you a take-it-or-leave-it offer. If you accept, then congratulations: you've found your supervisor! If you decline, then your jilted potential supervisor will be so heartbroken that they will be emotionally incapable of speaking to you again. This limits your ability to shop around for a supervisor. Once you've turned someone down, you can't go back and change your mind. Once you've accepted an offer, you can't renege.

Being a methodical but extremely busy young economist, you decide on the following two-phase strategy. In phase one you randomly choose an "initial sample" of `k` from the full list of `n` potential supervisors. You then meet with each of these `k`. No matter how well the meetings go, you *decline* these first `k` offers of supervision. At the end of this process you are still without a supervisor but you've collected `k` "supervisor scores." Call the maximum score among the initial sample `max_score`. In phase two, you sequentially meet with the remaining `n - k` supervisors in random order until you either encounter one whose score is higher than `max_score` or run out of options. In the latter case, you're stuck with whichever supervisor you met with last.

With the appropriate choice of `k`, this two-phase strategy can be shown to maximize the probability that you will end up with your most preferred supervisor. So what value of `k` should you choose? If you choose a very large value, there's a good chance that your most preferred supervisor will be among the initial `k`, so you'll turn down their offer. If you choose a very low value, then `max_score` will likely be too low and you'll end up "settling" for a less-than-ideal supervisor. It seems clear that the best choice of `k` should depend on `n`, but how? And if you make the best possible choice, when is the probability that you'll end up with your most preferred supervisor? Let's find out!

1. Write a function called `supervisor_sim()` that takes two input arguments: `n` is the number of potential supervisors, and `k` is the size of your phase one "initial sample." Your function should simulate one "draw" of the two-phase procedure described above:
   - Randomly permute the list of supervisors.
   - Identify the best supervisor from the initial sample of `k`.
   - Cycle through the remaining `n - k` supervisors until you hit your stopping criterion or run out of candidates.
   - Return the *rank* of the supervisor you end up with, i.e. `1` for your most preferred, `2` for your second most preferred, and so on.
2. Write a function called `get_prob_preferred()` uses `supervisor_sim()` to approximate the probability of getting your preferred supervisor for a given choice of `k` and `n` based on 10,000 simulation replications. Test this function with `n = 50` and `k = 5`. You should get a result between `0.23` and `0.26`. If you don't, then double-check your code for both this part and the preceding one.
3. I'll let you in on a secret: `k = 10` is *sub-optimal* for `n = 50` but the optimal value lies between `5` and `25`. Use simulation to approximate the optimal choice of `k` and the probability of getting your most preferred supervisor when `k` is chosen optimally. You may need to increase the number of simulation replications to help you zero in on the optimal choice.

Hints / Pep Talk: This is a *hard* problem, but once you crack the first part, the rest of the problem should be fairly straightforward. As usual when it comes to simulations, there are various ways to set this up, but one point may be helpful to consider. The ideas of "supervisor scores" and `max_score` in the write-up above are helpful for describing the stopping rule, but you do not necessarily need to generate and work with supervisor scores in your simulation code. It turns out to be enough to work with the *ranks* of the randomly-ordered supervisors. You may find it helpful to consult the help file for the base R function `which()`.