The Secretary Problem

Sebastien Plaasch
Maxime Rubio
Yannick Verbeelen

supervised by
Arturo Jaramillo Gil

December 29, 2020

Experimental Mathematics 2
University of Luxembourg
Faculty of Science, Technology and Medecine
Academic yaer 2020-2021 (Winter Semester)

Contents

1 Introduction

The *Secretary Problem* is a famous riddle in which an employer wants to hire a secretary but there a certain rules[1]:

1. There is a single position to fill.
2. There are n applicants for the position, and the value of n is known.
3. The applicants, if seen altogether, can be ranked best to worst unambiguously.
4. The applicants are interviewed sequentially in random order, with each order being equally likely.
5. Immediately after an interview, the interviewed applicant is either accepted or rejected, and the decision is irrevocable.
6. The decision to accept or reject an applicant can be based only on the relative ranks of the applicants interviewed so far.
7. The objective of the general solution is to have the highest probability of selecting the best applicant of the whole group. This is the same as maximizing the expected payoff, with payoff defined to be one for the best applicant and zero otherwise

The problem then is to find out when the employer should stop interviewing applicants to maximize the probability of hiring the best candidate. This report is going to show our way to finding the best possible strategy to this problem. While this problem has been studied thoroughly already, it is important to note that we first try to find solutions by ourselves before relying on past documentations.

First, we will try to find a strategy that we will improve until we have the best possible one. We will start by running some simulations of the problem, applying those strategies, and see how successful they are. Afterward we will find a formula to calculate the probability of success of our strategy depending on the number of applicants and on the amount of applicants that we "ignore". (This notion of ignoring applicants will be explained in the first section when we present our way of approaching the problem.) Then, we will approach our formula with a function that we will be able to optimize. This will give us the best possible probability of reaching our goal.

2 Policies

In this section, we will start by finding a simple strategy that we will keep on updating to improve, step by step, the probability of hiring the best applicant for the position. This part will mostly be based on intuition and experimentation.

2.1 Hiring the first candidate

The first policy that we considered is somewhat trivial: we hire the first applicant interviewed for the position. While this policy is not expected to deliver particularly interesting results by itself, it offers a fine introduction to the subject and opportunities for improvement.
In this scenario, we have n applicants for the position. The probability that we hire the best applicant is the probability that the best applicant is interviewed first: that is $1/n$. This is also the probability of hiring a candidate at random.
This has been experimentally confirmed via Monte Carlo simulations, with one hundread million simulations for different values of n. In this experiment, we measured the number of times the best candidate was

[1] via https://en.wikipedia.org/wiki/Secretary_problem

recruited over 10,000 simulations. We replicated this experiment 10,000 times for different values of n. Further details regarding how this simulation was implemented is available in the appendix A.1.1, page 15.

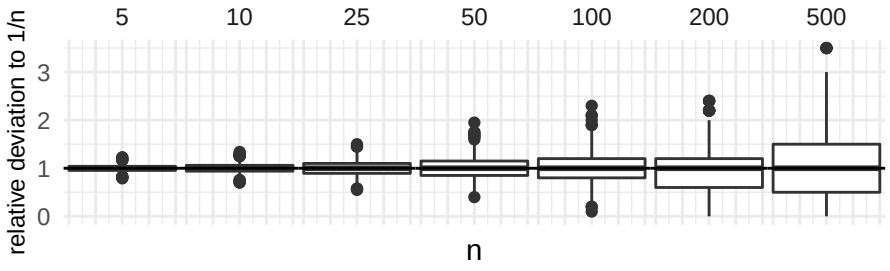

Figure 1: Boxplots displaying the relative deviation between average success rate of 10,000 simulations.

The relative deviation in the previous figure is calculated by dividing the recorded number of successes in a simulation by the expected value $1/n$. For example, if $n = 100$, the expected number of successes for 10,000 simulations is $10000 \cdot 1/100 = 100$. If the result of one simulation is 98, we get $98/100 = 0.98$. A value close to 1 means that the result of the simulation is close to the expected value.
The boxplots in the figure are well centered around 1, showing that there is close to no deviation around expected value.

2.2 Skipping the first applicant

Can we alter this initial policy to increase the probability of hiring the best applicant? We can do so by "skipping" the first candidate, and then continue the interview process, discarding any subsequent applicant deemed less suitable than the first one, and hire the next best candidate.
Intuitively, this is an improvement because the chance that the best applicant is the first one interviewed is low: $1/n$. It is therefore unlikely that we would discard the best applicant. Such policy also rids us of a certain number of applicants, which are by construction less suitable than the first applicant and by extension less suitable than the best applicant.
Simulations for this policy shows a noticeable improvement to the initial policy as shown in the following graphic:

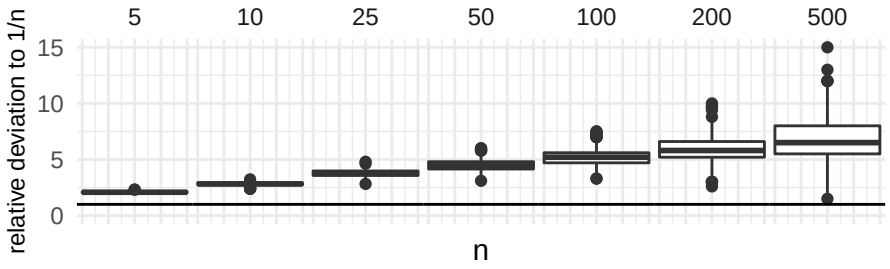

Figure 2: Boxplots displaying the relative deviation between average success rate of 10,000 simulations and $1/n$.

We see in this graphic that the relative deviation is consistently above 1, indicating that the probability is greater than for the first policy. Moreover, the improvement is even greater as n increases.

2.3 Skipping the r first applicants

We can reiterate the solution previously described in order to ignore the first two candidates, rather than the first one only. Indeed, it seems that doing so is still an improvement as the probability of the best candidate being one of the first two interviewed remains low: $2/n$. Moreover, should the second candidate be better than the first one, the threshold that we implement will be higher; allowing us to eliminate a greater number of candidates!
It seems fair to assume that we would continue to improve the probability of hiring the best candidate by discarding the third candidate, the fourth, and so on. As long as the gain in probability is greater than the loss caused by the increasing likelihood of ignoring the best candidate, this solution seems applicable.
However, there will be a point at which this gain in probability will not be sufficient to counter the loss caused by ignoring the next applicant. Can we find this?
We want to observe how the number r of ignored applicants changes the probability of hiring the best one. With this purpose we implemented a function, explained in detail in A.1.3 on page 17, that simulates the experiment and computes the ratio between successes and the number of simulations.
We made graphs representing the function for $n = 50$ and $n = 100$:

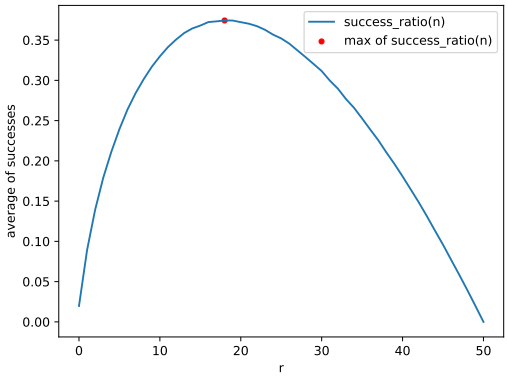

Figure 3: Graph of *success_ratio* for $n = 50$

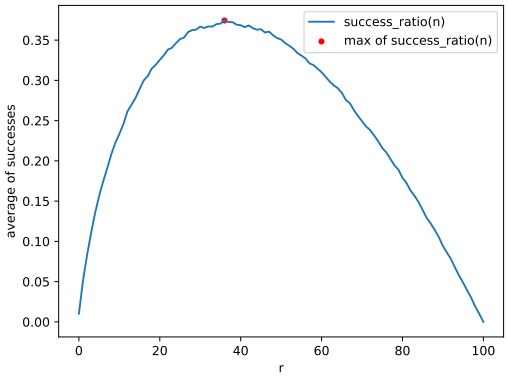

Figure 4: Graph of *success_ratio* for $n = 100$

We can clearly observe that there is a value for r where the ratio is maximal and for both graphs the function has the same behavior. We compute the maximum for each function: For $n = 50$ the maximum is reached at $r = 18$ and for $n = 100$ when $r = 36$. Furthermore the ratio at those maximums is $0,37439$ in both cases.

3 Measuring the probabilities

In this section, we will start by developping an explicit formula to measure the probability of hiring the best candidate. We will perform some exploratory analysis and issue some observations. After this, we will analyse the function to confirm these observations.

3.1 Finding an explicit formula

3.1.1 An explicit formula

Now we will try to find an explicit formula to calculate the probability of hiring the best applicant for a given number of applicants n depending on the amount of ignored candidates r. This will give us a function of r for a fixed integer n.

So we have n different applicants, which we can interview in any order. Therefore we can modelize this scenario as a vector $X = (x_1, \ldots, x_n)$ such that $x_k \in \{1, ..., n\}; \forall k \in \{1, \ldots, n\}$ and $x_i \neq x_j; \forall i \neq j$. There are $n!$ such vectors. We consider x_i to be a better candidate as x_j if $x_i > x_j$. So $n = \max_{i \in \{1,\ldots,n\}} \{x_i\}$ is the best candidate. Let p be the position of n (i.e. $x_p = n$) and r be the number of ignored candidates. To succeed, we need $p > r$, that is $r + 1 \geq p \geq n$. For any position p of our best candidate, we can extract two subvectors:

- $(x_1, \ldots, x_{p-1}) =: A$
- $(x_{p+1}, \ldots, x_n) =: C$

So that we have: $(\overbrace{x_1, \ldots, x_{p-1}}^{A}, \overbrace{x_p}^{n}, \overbrace{x_{p+1}, \ldots, x_n}^{C})$ We denote $m := \max\{x_i | x_i \in A\}$ the best candidate interviewed before x_p. Let's then divide A into two different subvectors:

- $(x_1, \ldots, x_r) =: I$, the ignored candidates
- $(x_{r+1}, \ldots, x_{p-1}) =: B$, the other ones

This gives us the following situation: $(\underbrace{x_1, \ldots, x_r}_{I}, \underbrace{x_{r+1}, \ldots, x_{p-1}}_{B}, \underbrace{x_p}_{n}, \underbrace{x_{p+1}, \ldots, x_n}_{C})$.

To succeed in hiring the best candidate, we need $m \in I$, else we would have $x_k = m > \max\{x_i | x_i \in I\}$ for a ceratin $k \in \{r + 1, \ldots, p - 1\}$ and we would thus hire $x_k \neq x_p$. This means that m can be in any of the r first positions.
However for any position of m, we can construct a vector $Y := (y_1, \ldots, y_{r-1})$ with:
$y_k = \begin{cases} x_{k+1} \; if \; m \in \{x_1, \ldots, x_k\} \\ x_k \; otherwise. \end{cases}$

The benefit of doing so is that we can define an equivalence relation over the vectors in consideration, by stating that two vectors of length r are in relation if their associated vectors of length $r - 1$ are the same.
For example: $X_1 := (x_1, x_2, m, x_3) \sim (m, x_1, x_2, x_3) =: X_2$ because $Y_1 = (x_1, x_2, x_3) = Y_2$.

We do not need to worry about the exact position of m in I, but only about the order of the other elements. Each of those vectors of length $r - 1$ will account for r different vectors of length r.
There exists P_{m-1}^{r-1} such vectors of length $r - 1$ for any given m. (Where $P_b^a = \frac{b!}{(b-a)!}$ is the number of permutations of a elements among b elements.) This gives us $r \cdot P_{m-1}^{r-1}$ different possibilities for I.
Then, for $(x_{r+1}, \ldots, x_{p-1})$, given $(x_1, \ldots, x_r)$, any arrangement of $p - r - 1$ elements of the remaining $m - r$ elements will do: Thus giving P_{m-r}^{p-r-1} possibilities for each $(x_1, \ldots, x_r)$.
So for any given m and p we can compute the number of arrangements for $(x_1, \ldots, x_{p-1})$ guaranteeing success by: $(r \cdot P_{m-1}^{r-1} \cdot P_{m-r}^{p-r-1})$.
For $(x_{p+1}, \ldots, x_n)$, given $(x_1, \ldots, x_p)$, any arrangement of $(n-p)$ elements, of the remaining $(n-p)$ elements will do: So we have $P_{n-p}^{n-p} = (n - p)!$
Therefore, the probability of success of hiring the best candidate is given by:

$$\begin{aligned} & \frac{1}{n!} \sum_{p=r+1}^{n} \sum_{m=p-1}^{n-1} r \cdot P_{m-1}^{r-1} \cdot P_{m-r}^{p-r-1} \cdot (n-p)! \\ = \; & \frac{r}{n} \sum_{p=r+1}^{n} \frac{(n-p)!}{(n-1)!} \sum_{m=p-1}^{n-1} \frac{(m-1)!}{(m-p+1)!} \end{aligned}$$

3.1.2 A simpler formula

In the paper *Who solved the secretary problem?* [1], the simpler formula $\frac{r}{n}\sum_{p=r+1}^{n}\frac{1}{p-1}$ for success probability is proved.
We will demonstrate that the formula we developed above is equivalent.
We have to prove the following :

$$\frac{r}{n}\sum_{p=r+1}^{n}\frac{(n-p)!}{(n-1)!}\sum_{m=p-1}^{n-1}\frac{(m-1)!}{(m-p+1)!}=\frac{r}{n}\sum_{p=r+1}^{n}\frac{1}{p-1}$$

In order to do that, we will prove, by induction over n, that $\forall\ n\in\mathbb{N}, \forall\ p\in\mathbb{N}$ such that $2\leq p\leq n$:

$$\frac{(n-p)!}{(n-1)!}\sum_{m=p-1}^{n-1}\frac{(m-1)!}{(m-p+1)!}=\frac{1}{p-1} \tag{1}$$

1. Consider the base case, $n=2$. Note that this implies that $p=2$ as well. Then, we have:

$$\begin{aligned}\frac{(n-p)!}{(n-1)!}\sum_{m=p-1}^{n-1}\frac{(m-1)!}{(m-p+1)!} &= \frac{0!}{1!}\sum_{m=1}^{1}\frac{(m-1)!}{(m-p+1)!}\\ &= 1\\ &= \frac{1}{p-1}\end{aligned}$$

2. Now, suppose that for $n\in\mathbb{N}$ the equation 1 is true and let $p\in\mathbb{N}$, $p\leq n$. Therefore, we have to prove that:

$$\frac{(n+1-p)!}{(n)!}\sum_{m=p-1}^{n}\frac{(m-1)!}{(m-p+1)!}=\frac{1}{p-1}$$

$$\Leftrightarrow\quad \frac{n+1-p}{n}\left(\frac{(n-p)!}{(n-1)!}\sum_{m=p-1}^{n-1}\frac{(m-1)!}{(m-p+1)!}+\frac{(n-p)!}{(n-1)!}\frac{(n-1)!}{(n+1-p)!}\right)=\frac{1}{p-1}$$

$$\Leftrightarrow\quad \frac{n+1-p}{n}\left(\frac{1}{p-1}+\frac{(n-p)!}{(n+1-p)!}\right)=\frac{1}{p-1}$$

$$\Leftrightarrow\quad \frac{n+1-p}{n}\left(\frac{1}{p-1}+\frac{(n-p)!}{(n+1-p)(n-p)!}\right)=\frac{1}{p-1}$$

$$\Leftrightarrow\quad \frac{n+1-p}{n(p-1)}+\frac{1}{n}=\frac{1}{p-1}$$

$$\Leftrightarrow\quad \frac{n+1-p+p-1}{n(p-1)}=\frac{1}{p-1}$$

Which proves the claim.

3.2 Analyzing the probability function

We will now perform exploratory analysis to observe the behaviour of the probability function. Let's start by formalizing some of the notation.

Let $n \in \mathbb{N}$ be the number of applicants for the position. Let $r \in \{1, ..., n-1\}$ be the number of ignored applicants for the interview. We define as $p(n, r)$ the probability of hiring the best of n candidates, discarding r of them.
We introduce the following functions:

$$\pi(n) := \max\{p(n, r) \mid r \in \{1, ..., n-1\}$$
$$\beta(n) := \min\{r \in \{1, ..., n-1\} \mid p(n, r) = \pi(n)\}$$

The function π returns, for a given n, the highest probability of hiring the best candidates by discarding some of them. β returns the number of candidates we should skip in order to maximize this probability. The decision of using the minimum is driven by the fact that there might be multiple r returning the same probability. It is therefore safer to consider the *min* (using max instead of min would have provided similar results).
We could summarize that by the following equality: $\pi(n) = p(n, \beta(n))$.
Let us start by reviewing the behaviour of the probability $\pi(n)$ as n increases.

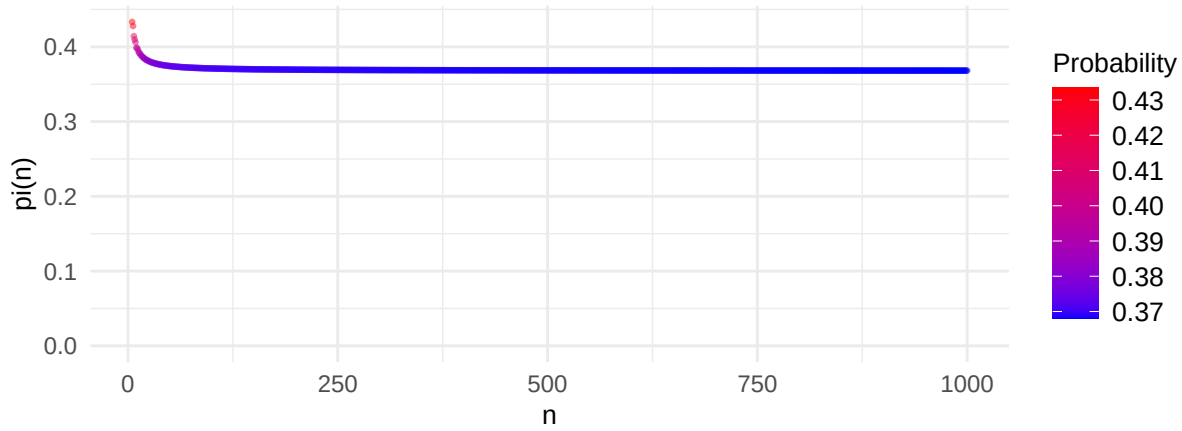

Figure 5: $\pi(n)$ for $n \in \{5, ..., 1000\}$

The graph of $\pi(n)$ shows that the probability of hiring the best candidate decreases when n increases. It however appears that this probability may be converging.

Now, let us observe how the optimal number of discarded candidates $\beta(n)$ evolves with n.

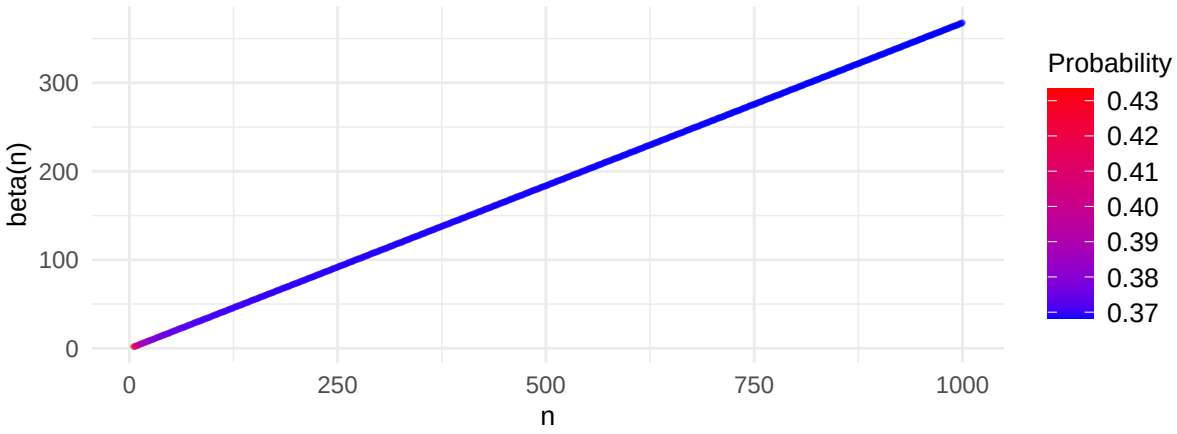

Figure 6: $\beta(n)$ for $n \in \{5, ..., 1000\}$

The graph of $\beta(n)$ shows that β is an increasing function, and seems linear, which indicates that the optimal numbers of ignored candidates increases proportionally to the number of candidates n.
The next figures shows a normalization of β, $n \mapsto \frac{\beta(n)}{n}$, which represents the ratio of discarded candidates for a given n (recall that $1 \leq \beta(n) < n$).

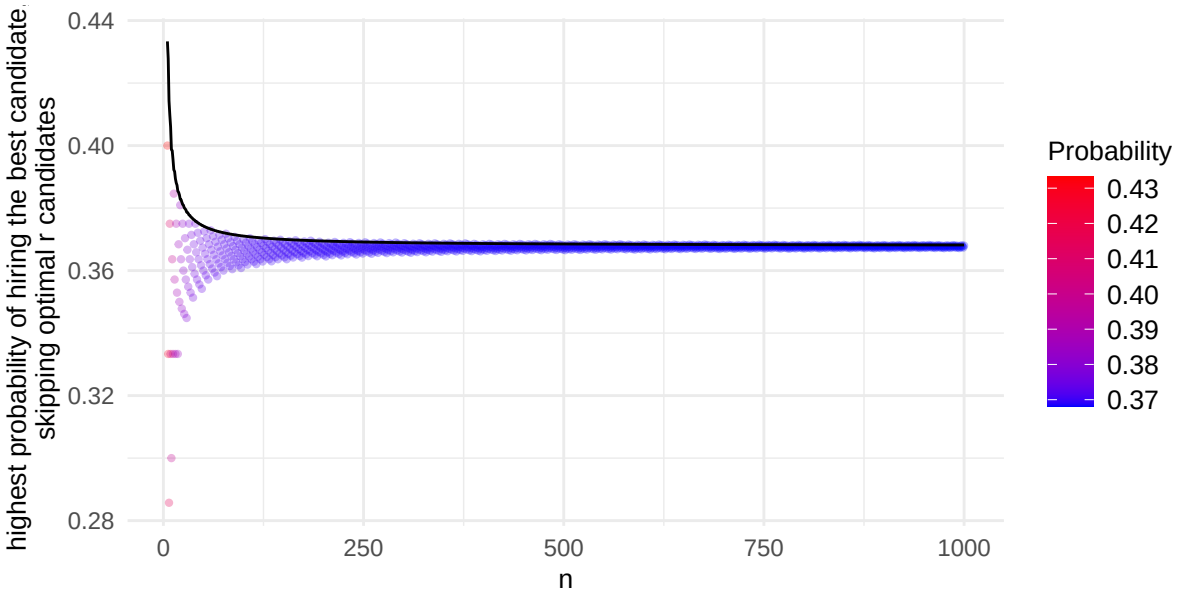

Figure 7: $\pi(n)$ (black line) and $\beta(n)/n$, $n \in \{5, ..., 1000\}$

This new figure shows that the normalized ratio $\beta(n)/n$ seems to be converging as well. This is an indication that there may exist an optimal ratio such that when $r \approx \lim_{n\to\infty} \frac{\beta(n)}{n}$, the probability $p(n, r)$ is maximal. It is also interesting to notice that $\frac{\beta(n)}{n}$ and $\pi(n)$ seem to converge towards the same limit.

3.3 What the formula tells us

We have seen in the previous section the convergence of the probability function and the ideal ratio. Hence, in this section, we will optimize the probability function to get the exact value of the convergence.

3.3.1 Approximation of the sum by an integral

Our goal here is to justify that we can approximate the sum $\frac{r}{n}\sum_{p=r+1}^{n}\frac{1}{p-1}$ by the integral $\frac{r}{n}\int_{r+1}^{n+1}\frac{1}{p-1}dp$, when $n \in \mathbb{N}$ is "big enough". Indeed, the sum can be considered as a Riemann sum with constant step 1. We will prove that even if the step of the Riemann sum doesn't get infinitely closer to 0 as $n \to \infty$, we can always choose $n \in \mathbb{N}$ such that $\left|\frac{r}{n}\sum_{p=r+1}^{n}\frac{1}{p-1} - \frac{r}{n}\int_{r+1}^{n+1}\frac{1}{p-1}dp\right| < \epsilon\,,\ \forall\ \epsilon > 0\,,\ \forall\ r \in \{1,\dots,n\}$.

1. First, we will prove that $\forall\ n \in \mathbb{N},$ with $n > 1, \forall\ r \in \{1,\dots,n\}$:

$$\sum_{p=r+1}^{n}\frac{1}{p-1} - \int_{r+1}^{n+1}\frac{1}{p-1}dp < \frac{1}{r} - \frac{1}{n} \tag{2}$$

Let $n \in \mathbb{N},$ with $n > 1$ then, $\forall\ r \in \{1,\dots,n\}$:

$$\frac{1}{r} - \int_{r+1}^{r+2}\frac{1}{p-1}dp = \frac{1}{r} - \log\left(\frac{r+1}{r}\right) < \frac{1}{r} - \frac{1}{r+1}$$

Let $r \in \{1,\dots,n\}$. Then:

$$\begin{aligned}\sum_{p=r+1}^{n}\frac{1}{p-1} - \int_{r+1}^{n+1}\frac{1}{p-1}dp &= \frac{1}{r} - \int_{r+1}^{r+2}\frac{1}{p-1}dp + \frac{1}{r+1} - \int_{r+2}^{r+3}\frac{1}{p-1}dp + \dots \\ &\quad \dots + \frac{1}{n-1} - \int_{n}^{n+1}\frac{1}{p-1}dp \\ &< \frac{1}{r} - \frac{1}{r+1} + \frac{1}{r+1} - \frac{1}{r+2} + \dots + \frac{1}{n-1} - \frac{1}{n} \\ &= \frac{1}{r} - \frac{1}{n}\end{aligned}$$

2. Furthermore, we will show that $\frac{r}{n}\sum_{p=r+1}^{n}\frac{1}{p-1} - \frac{r}{n}\int_{r+1}^{n+1}\frac{1}{p-1}dp \xrightarrow[n\to\infty]{} 0$. In fact $\forall\ r \in \{1,\dots,n\}$:

$$(2) \Rightarrow \frac{r}{n}\sum_{p=r+1}^{n}\frac{1}{p-1} - \frac{r}{n}\int_{r+1}^{n+1}\frac{1}{p-1}dp < \frac{1}{n} - \frac{r}{n^2} \leq \frac{1}{n} - \frac{1}{n^2} \tag{3}$$

where $\lim_{n\to\infty}\frac{1}{n} - \frac{1}{n^2} = 0.$

$$\Rightarrow \lim_{n\to\infty}\left(\frac{r}{n}\sum_{p=r+1}^{n}\frac{1}{p-1} - \frac{r}{n}\int_{r+1}^{n+1}\frac{1}{p-1}dp\right) = 0$$

$$\Rightarrow \exists N \in \mathbb{N}\ \text{s.t.}\ \forall n > N : \left|\frac{r}{n}\sum_{p=r+1}^{n}\frac{1}{p-1} - \frac{r}{n}\int_{r+1}^{n+1}\frac{1}{p-1}dp\right| < \epsilon\,,\ \forall\ \epsilon > 0$$

Which proves that we can find $n \in \mathbb{N}$, sufficiently big, such that:

$$\frac{r}{n}\sum_{p=r+1}^{n}\frac{1}{p-1} \approx \frac{r}{n}\int_{r+1}^{n+1}\frac{1}{p-1}dp.$$

3.3.2 Optimization of the integral

Now, let $n \in \mathbb{N}$ and consider a function $r \mapsto \frac{r}{n}\int_{r+1}^{n+1} \frac{1}{p-1}dp = -\frac{r}{n}\log\left(\frac{r}{n}\right)$. We can derivate this function to find its extremums:

$$\left(-\frac{r}{n}\log\left(\frac{r}{n}\right)\right)' = -\frac{1}{n}\log\left(\frac{r}{n}\right) - \frac{1}{n} = -\frac{1}{n}\left(\log\left(\frac{r}{n}\right)+1\right)$$

To find the extremums, we have to solve $-\frac{1}{n}\left(\log\left(\frac{r}{n}\right)+1\right) = 0$.

$$\begin{aligned} &\Leftrightarrow \quad \log\left(\frac{r}{n}\right)+1 &&= 0 \\ &\Leftrightarrow \quad \frac{r}{n} &&= \frac{1}{e} \\ &\Leftrightarrow \quad r &&= \frac{n}{e} \end{aligned}$$

By studying the sign of the derivative, we can conclude that $r \mapsto \frac{r}{n}\int_{r+1}^{n+1} \frac{1}{p-1}dp$ reaches its maximum when $r = \frac{n}{e}$. Moreover, the value of the maximum is :

$$-\frac{1}{e}\log\left(\frac{1}{e}\right) = \frac{1}{e}$$

3.3.3 Application to the sum

We can apply the previous results to conclude that the sum $\frac{r}{n}\sum_{p=r+1}^{n} \frac{1}{p-1}$ is maximized when $r \approx \frac{n}{e}$, i.e $r = \left\lfloor \frac{n}{e} \right\rfloor$ or $r = \left\lceil \frac{n}{e} \right\rceil$ and the result of the sum should be approximately $\frac{1}{e}$. This is even more precise as n is big. For example, the inequalities from (3) tell us that, for $n = 100$, $r = 36$ (optimal r for $n = 100$) the difference between the Riemann sum and the integral is smaller than $\frac{1}{100} - \frac{36}{100^2} = 0.0064$. In fact, the difference is 0.0032. Moreover, when $r' = 37$, we have $\left|\frac{100}{e} - r'\right| < \left|\frac{100}{e} - r\right|$, thus the optimal r is not necessarily the closest to $\frac{n}{e}$.

Fun fact :

$$\begin{aligned} &\text{If } r \approx \frac{n}{e} \text{ then } \frac{r}{n}\sum_{p=r+1}^{n} \frac{1}{p-1} &&\approx \frac{1}{e} \\ \Rightarrow \quad &\frac{1}{e}\sum_{p=r+1}^{n} \frac{1}{p-1} &&\approx \frac{1}{e} \\ \Rightarrow \quad &\sum_{p=r+1}^{n} \frac{1}{p-1} &&\approx 1 \end{aligned}$$

This is well explained by studying, for a given $n \in \mathbb{N}$, the sign of $p(n,r) - p(n,r+1)$.

$$\begin{aligned} p(n,r) - p(n,r+1) &= \left(\frac{r}{n}\sum_{p=r+1}^{n} \frac{1}{p-1}\right) - \left(\frac{r+1}{n}\sum_{p=r+2}^{n} \frac{1}{p-1}\right) \\ &= \frac{r}{n}\left(\sum_{p=r+1}^{n} \frac{1}{p-1} - \sum_{p=r+2}^{n} \frac{1}{p-1}\right) - \frac{1}{n}\sum_{p=r+1}^{n} \frac{1}{p-1} \\ &= \frac{r}{n}\cdot\frac{1}{r} - \frac{1}{n}\sum_{p=r+2}^{n} \frac{1}{p-1} \\ &= \frac{1}{n} - \frac{1}{n}\sum_{p=r+2}^{n} \frac{1}{p-1} \end{aligned}$$

The sign of this difference depends only on the value of $r \mapsto \sum_{p=r+2}^{n} \frac{1}{p-1}$, which is strictly decreasing. The probability increases until $\sum_{p=r+2}^{n} \frac{1}{p-1} > 1$, and starts to decrease afterwards (confirming the shape observed in figures 3 and 4). The optimal r is $\min\{r \mid \sum_{p=r+2}^{n} \frac{1}{p-1} \leq 1\}$, confirming the observation.

3.4 The optimal integer

We know that n/e is not an integer, while the number of discarded candidates must be one. This raises the following question: which $r \in \mathbb{N}$ gives the actual best probability? Is it the closest integer to n/e? the one above? the one below? To figure this out, we will start by plotting $r - n/e$, and observe how the optimal r behaves (see figure 8).

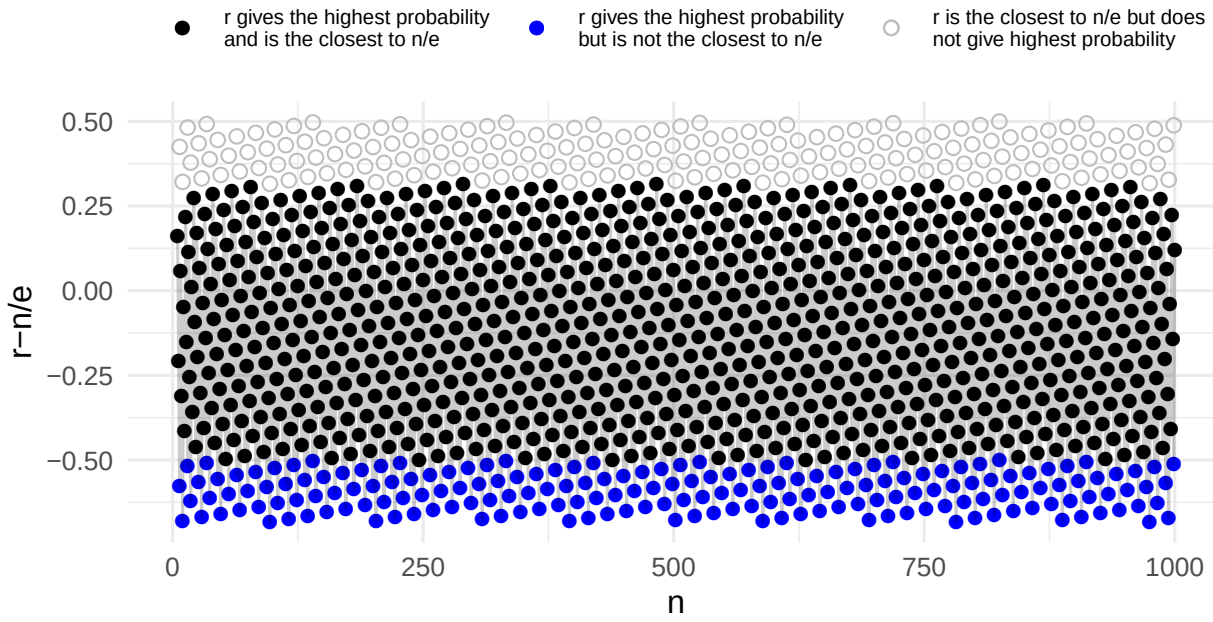

Figure 8: Comparison between the optimal r and n/e

The plot shows that the optimal r is neither the closest integer, the ceiling or the floor of n/e: it can be either one of them, and is sometimes not the closest integer. We can identify a threshold, located around $+0.31$ after which, even though r is the closest integer, it is not the optimal number of skipped candidates. Moreover, the concentration of such points does not appear to decrease when n increases.
Let us shade the surfaces in which (1) r maximizes the probability and is the closest integer to n/e, (2) r maximizes the probability but is not the closest integer to n/e and, (3) r is the closest integer but does not maximize the probability (see figure 9).

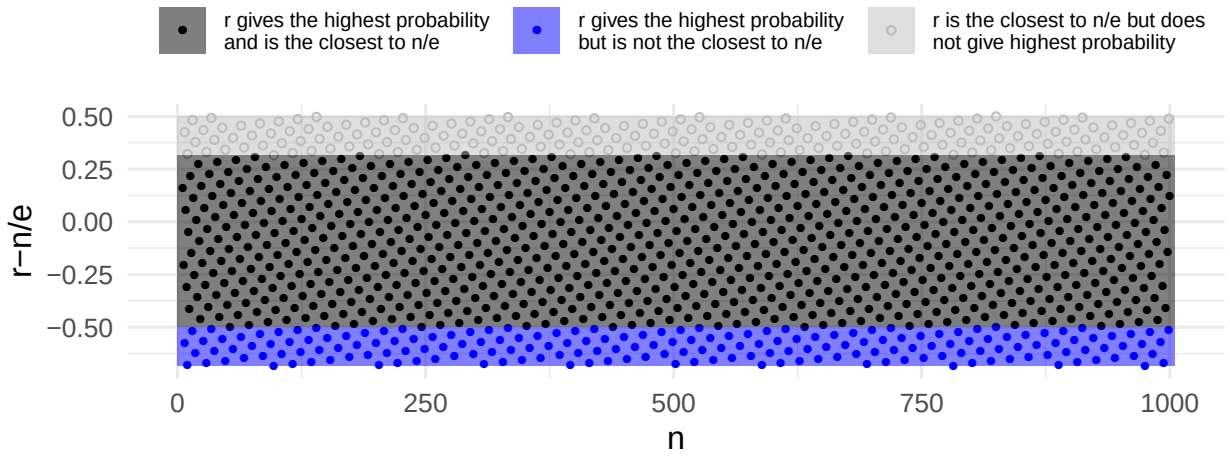

Figure 9: Surfaces representing different types of r

Note that available data for $5 \leq n \leq 1000$ does not show any overlap between those surfaces, nor any evidence that n has any influence on these. Based on this information, the optimal r lays between $n/e+0.35$ and $n/e-0.684$. The blue and light gray surfaces in figure 9 appear to be the same height, indicating that we can correct n/e, altering it by subtracting a constant c.
We can try to get an approximation of this constant by computing the average of these errors: $c \approx \frac{0.315-0.684}{2} = -0.1845$. The ratio n/e seems to be around 0.1845 too high compared to the optimal ratio.
We can verify this again by correcting our calculation by computing $r-(n/e-0.1845)$:

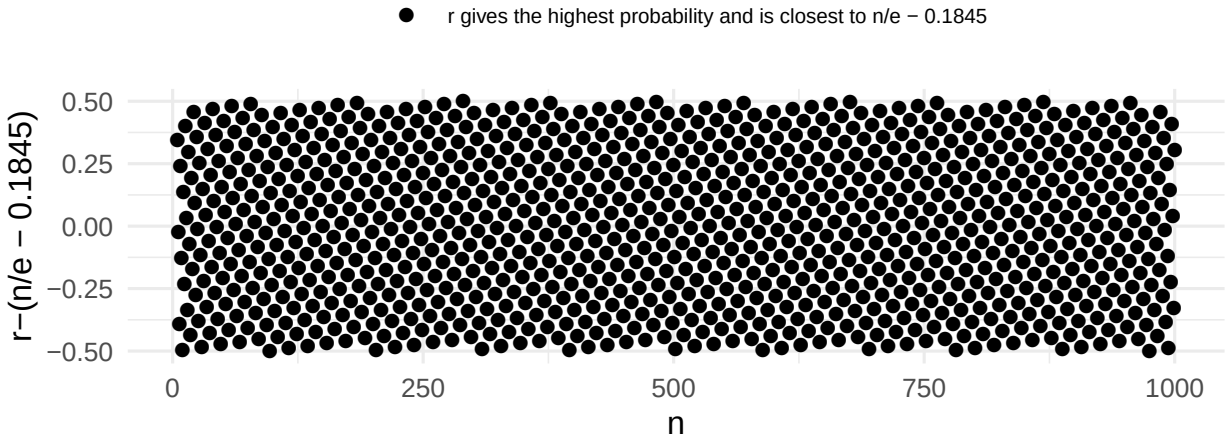

Figure 10: Correction by comparing optimal r to $(n/e-0.1845)$

In this case, we always get that the optimal r is indeed the closest integer to $n/e-0.1845$.
Additionally we have found that this is not true for a few $n \geq 1000$. (only 3 cases for $n \leq 10000$) However with the same approach it may be possible to find a preciser value (close to 0.1845) such that the closest integer to n/e minus that new value would be the optimal r for $n \leq 10000$ for example. Sadly we were not able to find a prove for this working endlessly and neither if there exists one fixed correction to the ratio n/e

that would work for all $n \in \mathbb{N}$.

4 Conclusion

By trying out different policies, we quickly realised that the best strategy would be to ignore a certain amount of applicants and hire the next best one. This only left the question: What is the ideal amount of ignored applicants?
To answer this question we started by finding a formula for the probability of hiring the best candidate depending on the total of applicants and the number of ignored ones. We then approximated this formula by a function that can be optimized thus giving us the optimal ratio of ignored candidates. This ratio is precisely n/e where n is the amount of applicants.
Finally we tried to find an ideal integer or at least a value such that the closest integer to that value would be the optimal number of ignored candidates since this isn't always the case for the ratio n/e. We found that for $n \leq 1000$ the ideal integer is always the one closest to $n/e - 0.1845$. However we have seen that this does not work for every $n \geq 1000$ and neither were we able to prove if there even exists a universal correction of the ratio n/e that would work for every $n \in \mathbb{N}$, this would be a nice lead for further research on this project.
Nonetheless if somebody would see himself in this exact situation in real life, it is important to note that his best bet of success would be to just take either one of the two closest integers of n/e since the difference of their probability of success becomes minimal as n increases.

References

[1] Thomas S. Ferguson. Who solved the secretary problem? *Statistical Science*, 4(3):282–289, August 1989.

Appendixes

A Code

A.1 Simulations

A.1.1 Simulation for hiring the first candidate

The experience is described as follow: for a given n, we generate a vector of length n obtained as a uniform permuation of the integers between 1 and n. Taking 1 as the best candidate, we verify if the first value of this vector is 1.
We then compute this simulation 1000 times, and measure the number of success. This measure is taken 1000 times, and the average is reported. The simulation is therefore computed 1,000,000 times for a given n.

../code/HireFirstApplicant.R

```r
library(magrittr)
library(dplyr)

set.seed(0.5577001) # ensures that re-run will generate same results.

#Generate an empty CSV to record simulations
write.csv(data.frame(n=integer(),
                     simulations=integer(),
                     positive=integer()),
          "../data/HireFC2.csv",row.names = F)

#Single simulation
policy <- function(k) {
  applicants = sample(k,k,replace=F)
  applicants[1] == 1 # returns true if first element is 1, false otherwise
}

#Generates a single output for 1000 single simulations
tryPolicy=function(n){
  #Generate an output for 1000 simulations.
  result = replicate(1000, policy(n))
  tibble(n,simulations=1000,positive=sum(result , na.rm=T))
}

#Populate CSV the average of 1000 simulations
tryAgain = function(n){
  fn = function(n){
    tryPolicy(n) %>% write.table(.,"../data/HireFC2.csv",
                                 row.names = F,
                                 sep=",",
                                 col.names = F,
                                 append = T)
  }
  replicate(1000,fn(n))
}

sapply(c(5,10,25,50,100,200,500),tryAgain)
```

A.1.2 Simulation for skipping the first candidate

We modified the previous program so that we first filter out the first candidate and any candidate with a score lower than the first one. We then verify if the first value of the filtered vector is 1.

../code/skip_FC.R

```
library(magrittr)
library(dplyr)

set.seed(0.5577001) # ensures that re-run will generate same results.

#Generate an empty CSV to record simulations
write.csv(data.frame(n=integer(),
                     simulations=integer(),
                     positive=integer()),
          "../data/Skip_FC2.csv",row.names = F)

#Single simulation
policy <- function(k) {
  applicants = sample(k,k,replace=F)
  bar = min(applicants[1:1])
  remainder= tail(applicants,k-1)
  remainder[remainder<bar][1] == 1 # returns true if first element is 1, false
      otherwise
}

#Generates a single output for 1000 single simulations
tryPolicy=function(n){
  #Generate an output for 1000 simulations.
  result = replicate(1000, policy(n))
  tibble(n,simulations=1000,positive=sum(result , na.rm=T))
}

#Populate CSV the average of 1000 simulations
tryAgain = function(n){
  fn = function(n){
    tryPolicy(n) %>% write.table(.,"../data/Skip_FC2.csv",
                                 row.names = F,
                                 sep=",",
                                 col.names = F,
                                 append = T)
  }
  replicate(1000,fn(n))
}

sapply(c(5,10,25,50,100,200,500),tryAgain)
```

A.1.3 Simulation for skipping the first r candidates

The function takes r as a parameter which is the number of ignored applicants and the applicants are denoted by an integer between 1 and n, where n is the total of applicants. The integer assimilated also represents the value of each applicant, so for example n is the best candidate, 1 the worst, etc.
In each simulation the function creates a list called, *candidates*, containing every integer from 1 to n, representing the applicants, then randomly chooses r integers and removes them from the list and stores them in another list named *interviewed*, standing for the ignored candidates.
Afterward the function checks if n belongs to the list *interviewed*, if this is the case the simulation is not a success. Else the function picks one integer, let's called it x, in the list *candidates* and compares it with all the integers from *interviewed*. If x is greater than all those integers the candidate x is selected, if not, the same procedure continues until the function finds an integer greater than all those in *interviewed*. The simulation is considered a success if the selected number is n.
Moreover the function runs a sufficient amount of simulations such that the ratio is as close as possible to the theoretical probability.
In this order the ratio is computed after each simulation and stored in a variable, called *intermediate_ratio*, and compare it with the ratio after the next simulation, named *ratio*. The comparison is done by computing the following distance : $d = \left| \frac{ratio}{intermediate\_ratio} - 1 \right|$. We suppose that the number of simulations is enough if $d < \epsilon = 10^{-6}$.

../code/success_ratio.py

```
1  import math
2  from random import choice
3  import matplotlib.pyplot as plt
4
5  N = 50
6  epsilon = 1e-6
7
8  def success_ratio(r):
9      """We create a list of N integers from 1 to N
10     representing the level of each candidate. Then
11     we randomly choose k integers in the list which are
12     removed from the list. Then we randomly choose one
13     integer in the list until we have an integer bigger
14     than the first k integers selected and consider it a success
15     if that integer is N. The returned value is the ratio of
16     successes over at least thousand experiments."""
17     if r == N:
18         return 0
19     n, success, absolute_err, ratio = 0, 0, 0, 1
20     while n < 1000 or math.fabs(absolute_err-1) > epsilon:
21         intermediate_ratio = ratio
22         n += 1
23         candidates = [i for i in range(1, N+1, 1)]
24         interviewed = []
25         for i in range(r):
26             r_candidate = choice(candidates)
27             candidates.remove(r_candidate)
28             interviewed.append(r_candidate)
29         if not(N in interviewed):
30             while len(candidates) > 0:
31                 candidate = choice(candidates)
32                 candidates.remove(candidate)
33                 if all([candidate > i for i in interviewed]):
```

```
34                      if candidate == N:
35                          success += 1
36                      break
37          ratio = success/n
38          if n >= 1000:
39              absolute_err = ratio/intermediate_ratio
40      return ratio
```

A.1.4 First attempt to calculate probability

This program generates a table (csv) computing the probability of hiring the best candidate for every n between 5 and 1000, and every r between 1 and $n-1$.

../code/proba_table.R

```r
try_r = function(r,n){
  # given r and n, computes the big sum and outputs a data.frame row
  proba = sapply(seq(r+1,n),FUN=function(p) 1/(p-1)) %>% sum(.) * r/n
  data.frame(n,r,proba)
}

try_all_r = function(n){
  # given n, computes the sum for all r from 1 to n-1, outputs a data frame row
  lapply(seq(1,n-1),FUN= function(r) try_r(r,n)) %>% do.call(rbind,.)
}

# Generate a table for all n between 5 and 1000.
tab = lapply(seq(5,1000),FUN=function(n) try_all_r(n)) %>% do.call(rbind,.)
write.csv(tab,"../data/proba_table.csv",row.names = F)
```

A.2 Visualisations

../code/boxplots.R

```r
library(ggplot2)
library(dplyr)

df_HFC <- read.csv("../data/HireFC2.csv")

df_HFC %>%
  mutate(avg = positive/simulations, deviation = avg/(1/n)) %>%
  ggplot() + geom_boxplot(aes(1,deviation),width=0.1) +
  geom_hline(yintercept=1) +
  facet_grid(cols=vars(n)) +
  theme(panel.spacing.x = unit(0, "lines"),
        axis.text.x = element_blank(),
        axis.line = element_blank(),
        axis.title.y = element_text(size=9)) +
  expand_limits(y = 0)+
  labs(x = "n",
       y = "relative deviation to 1/n") +
  scale_y_continuous() +
  ggsave("../data/image/Hire_FC.pdf",width=12,height=4,unit="cm")

df_SFC <- read.csv("../data/Skip_FC2.csv")

df_SFC %>%
  mutate(avg = positive/simulations, deviation = avg/(1/n)) %>%
  ggplot() + geom_boxplot(aes(1,deviation),width=0.1) +
  geom_hline(yintercept=1) +
  expand_limits(y = 0)+
  facet_grid(cols=vars(n)) +
  theme(panel.spacing.x = unit(0, "lines"),
        axis.text.x = element_blank(),
        axis.line = element_blank(),
        axis.title.y = element_text(size=9)) +
  labs(x = "n",
       y = "relative deviation to 1/n") +
  scale_y_continuous() +
  ggsave("../data/image/Skip_FC.pdf",width=12,height=4,unit="cm")
```

../code/Rplots.R

```r
library(magrittr)
library(ggplot2)
library(dplyr)

tab <- read.csv("../data/proba_table.csv")
theme_set(theme_minimal())
theme_update(axis.title = element_text(size=10),
             legend.title= element_text(size=10),
             legend.text= element_text(size=10))

tab %>%
  #Select highest proba for each n
  group_by(n) %>% top_n(1,proba) %>%
  ggplot(aes(n,proba,color=proba)) +
  geom_point(alpha=.5,size=.5) +
  scale_color_gradient(low="blue",high="red") +
  labs(x="n",
       y="pi(n)",
       color="Probability") +
  ylim(0,NA) +
  labs(color="Probability") +
  ggsave("../data/image/pi_n.pdf",height=6,width=16,units="cm")

tab %>% group_by(n) %>% top_n(1,proba) %>%
  ggplot() +
  geom_point(aes(n,r,color=proba), alpha=.5,size=.5) +
  scale_color_gradient(low="blue",high="red") +
  labs(x = "n",
       y = "beta(n)",
       color="Probability")+# scale_x_log10() +
  ggsave("../data/image/beta_n.pdf",height=6,width=16,units="cm")

  tab %>% #Select highest proba for each n
  group_by(n) %>% top_n(1,proba) %>%
  ggplot(aes(n,r/n,color=proba)) +
  geom_point(alpha=.5) + scale_color_gradient(low="blue",high="red") +
  labs(x="Number of candidates (n)",
       y="Highest probability pi(n)",
       color="Probability") +
    ggsave("../data/image/plot_points1.pdf")

tab %>% mutate(r = r/n) %>%
  #Select highest proba for each n
  group_by(n) %>% top_n(1,proba) %>% #pivot_longer(-n) %>%
  ggplot() +
  geom_point(aes(n,r,colour=proba),alpha=.3, size=.8) +
  geom_line(aes(n,proba)) +
  scale_color_gradient(low="blue",high="red") +
  ylab("highest probability of hiring the best candidate, \n skipping optimal r
      candidates") +
  xlab("n") +
  labs(color="Probability") +
  ggsave("../data/image/plot_points3.pdf", height=8,width=16,units="cm")
```

../code/Actual_Best_R.R

```r
library(dplyr)
library(ggplot2)
library(magrittr)
library(readr)
library(tidyr)

df <- read_csv("../data/proba_table.csv") #read data from the proba table

#Alter table
df2 <- df %>% group_by(n) %>%
  mutate(delta= r-n/exp(1), # r - n/e
                       abs_delta = abs(delta), # absolute difference
                       rk = as.character(rank(-proba)), # ranks by proba
                       rk2 = as.character(rank(abs_delta))) %>% # ranks by delta
  ungroup()

theme_set(theme_minimal())
theme_replace(legend.title = element_blank(),
              legend.position="top",
              legend.justification = "center",
              panel.spacing.x = unit(-2,"cm"),
              legend.text= element_text(size=7))
k = 1000 #upperbound of the plot
s = 2
labs = c("1"="r gives the highest probability \nand is the closest to n/e ",
         "2"="r gives the highest probability \nbut is not the closest to n/e",
         "3"="r is the closest to n/e but does \nnot give highest probability")
col = c("1"="black","2"="blue","3" ="gray")

ggplot() +
  geom_point(data = filter(df2, rk==1 & n <= k), aes(n,delta,color=rk2,shape=rk2),
      size=s) +
  geom_line(data = filter(df2, rk==1 & n <= k), aes(n,delta),alpha=.2) +
  geom_point(data = filter(df2, rk2 == 1 & rk!= 1 & n <= k), aes(n,delta,color="3",
      shape="3"),size=s) +
  labs(y="r-n/e") +
  scale_color_manual(name="Points",
                       labels=labs,
                       values=col) +
  scale_shape_manual(name="Points",
                       labels=labs,
                       values=c("1"=16,"2"=16,"3"=1)) +
  ggsave("../data/image/pointline.pdf",height=8,width=15,units="cm")

df3 <- rbind(
  filter(df2, rk == 1 & rk2 == 1) %>%
    summarize(class=1,
              m= min(delta),
              M = max(delta),
              color="black"),
  filter(df2, rk == 1 & rk2 != 1) %>%
    summarize(class=2,
              m= min(delta),
              M = max(delta),
              color="blue"),
```

```
54    filter(df2, rk != 1 & rk2 == 1) %>%
55      summarize(class=3,
56                m= min(delta),
57                M = max(delta),
58                color="gray")
59  )
60
61  ggplot() +
62    geom_point(data = filter(df2, rk==1 & n <= k), aes(n,delta,color=rk2,shape=rk2),
        size=s/2) +
63    geom_point(data = filter(df2, rk2 == 1 & rk!= 1 & n <= k), aes(n,delta,color="3",
        shape="3"),size=s/2) +
64    geom_rect(data=df3, aes(xmin=0, xmax=k+5,ymin=m,ymax=M, fill=as.character(class)),
        alpha=.5) +
65    labs(y="r-n/e") +
66    scale_fill_manual(name="Points",
67                      labels=labs,
68                      values=col) +
69    scale_color_manual(name="Points",
70                       labels=labs,
71                       values=col) +
72    scale_shape_manual(name="Points",
73                       labels=labs,
74                       values=c("1"=16,"2"=16,"3"=1)) +
75    ggsave("../data/image/areas.pdf",height=6,width=15,units="cm")
76
77  df4 <- df %>% group_by(n) %>%
78    mutate(delta= (r-(n/exp(1)-0.1845)), # r - n/e
79           abs_delta = abs(delta), # absolute difference
80           rk = as.character(rank(-proba)), # ranks by proba
81           rk2 = as.character(rank(abs_delta)))  # ranks by delta
82
83  df4
84
85  ggplot() +
86    geom_point(data = filter(df4, rk==1 & n <= k), aes(n,delta,color=rk2,shape=rk2),
        size=s) +
87    # geom_line(data = filter(df4, rk==1 & n <= k), aes(n, delta),alpha=.2) +
88    labs(y="r-(n/e - 0.1845)") +
89    scale_color_manual(name="Points",
90                       labels="r gives the highest probability and is closest to n/e -
                             0.1845",
91                       values=col) +
92    scale_shape_manual(name="Points",
93                       labels="r gives the highest probability and is closest to n/e -
                             0.1845",
94                       values=c("1"=16,"2"=16,"3"=1)) +
95    ggsave("../data/image/points01845.pdf",height=6,width=15,units="cm")
```