Secretary (Marriage) Problem

Hiroshi Hamada (ver04, 2016,1005)

花京院と青葉の文体練習 3.1

登場人物

神杉 青葉（かみすぎ　あおば）： S 大学文学部　数理行動科学研科 2 年生. 数学がちょっと苦手な大学生．父親の影響でファーストガンダムが好き

花京院 佑（かきょういん　たすく）: S 大学文学部　数理行動科学科 2 年生．数学が好きな大学生．スタンド能力はない.

注) 二人について，より詳しく知りたい方は『数理社会学入門—ベルヌーイ変奏曲』をご覧ください．http://www.sal.tohoku.ac.jp/~hamada/#text

---

秘書問題あるいは結婚問題

「ねえ，花京院君. 《運命の人》っていると思う？」神杉青葉の質問はかなり唐突だった．研究室のパソコンでシミュレーション用のコードを書いていた花京院は，ふうっとため息をつくと，青葉の方に目を向けた．

「君ねえ，僕だって，年中暇ってわけじゃないんだよ」

「うん．わかっているんだけど．こういうことって花京院君にしか聞けないんだよ．ホントに忙しかったら別の日でいいけど．ダメ？」

「あんまり僕が得意な話題じゃなさそうだけどな」

「えーっと，じゃあ言い換えると,『運命の人』に巡り会う確率を最大限高める方法は何か？　って質問だよ」

花京院の目が一瞬するどく光った.「そういうことか，それなら考えたことがある」

「やっぱり．こういうヘンなことを真剣に考えてるのって，花京院君しかいないだろなって思ったんだ」

「ヘンなこと，は余計だよ．それは，Martin Gardner　が 1960 年に Scientific American　のコラムで発表したと言われている有名な問題だ．秘書問題（Secretary Problem）って呼ばれてるけど，一般的には，どこにいるか分からないベストな対象を探す問題だから結婚問題 (Marriage Problem) とも言われている」

「へえ，けっこう昔から知られてる問題なんだね」

「イメージしやすいように，自分がこれから出会う人の中から一番いい人を見つけるにはどうしたらいいか，という問題として考えてみよう」

仮定

「それじゃあ，まず仮定の確認だよ」花京院はホワイトボードに仮定を書いた．

1. n 人とランダムに出会う

2. プロポーズできるのは 1 人だけ
3. n 人に対して完全に順序づけできる
4. 1 度逃した人は 2 度と現れない

仮定の意味

「n 人のなかから一番いい人を選ぶ，もっとも簡単な方法は，なんだと思う？」花京院が聞いた.

「『えーっと，全員観察して得点を記録しておいて，1 位の人を決める』じゃないかな」

「そうだね．それが一番簡単だね．ところが《仮定 4.》は『全員観察してから選べない』という制約を意味しているんだ」

「えー．そうなの？　なんで？」青葉が聞いた

「多分，競争の激しさを想定してるんだと思う．観察してる間に，別の人に取られちゃう状況を表現してるんじゃないかな」

「ふむふむ」

「だから，このモデルが考えている世界では，1 人 1 人と出会って，その都度，その人にプロポーズするかどうかを決めないといけないんだ．例えば賃貸アパートを見て回る状況なんかが似ているよ．アパートの内見をしているときに，ここはいい物件だから早く決めないと別の人が借りてしまうよって言われたことない？」

「あー，あるある．ああ言われるとあせるんだよねー．服なんかもそうだね．このサイズは今買わないと売り切れちゃうますよ，とかよく言われる.」

「恋愛市場も同じで，早く決めないと誰かにとられちゃうってことだよ．このモデルのおもしろいところは,《仮定 3.》より，n 人の中には，必ず（自分にとっての）1 位が存在するってとこなんだ．だから，この問題は「どこにいるか分からない 1 位の人」を探し出す問題とも考えることができる．少しロマンティックに表現すると，こんな感じだね」

> あなたがこれから出会う人の中に必ず「運命の人（1 位)」がいます．しかしその人が「運命の人」かどうかは，あなたには分かりません．あなたが「この人だ」と決めた時点で，運命の人探しが終わるからです．あなたが決めた後に，実は,「もっといい人」が控えているのかもしれません．あなたは運命の人に出会えるでしょうか？

「おー．ちょっとおもしろそうだね」青葉が身をのりだした．

問題

「さっき述べた条件で,『n 人の中にいるはずの 1 位』を探し当てる確率を最大化するにはどうしたらいいか？　これが考えるべき問題だよ．1 回だけしかできない選択で，間違わないようにするにはどうしたらいいか？　とも言えるね」花京院が条件を整理した．

「うわー．プレッシャーかかるー」

「いつものように，自分が当事者になったつもりで考えるんだよ．まず，最初に出会った人を選ぶかどうか考えてみよう．君ならどうする？　最初に出会った人に決める？」

「むむ……，もうちょっと，待ちたいかな」

「どうして？」花京院が聞いた

「いや，ほらいきなり決めるとなんかがっついてる 感じがしない？」

「もうちょっと論理的な理由で説明できないもんかなあ」花京院がため息をついた．

「だって，どうせなら 1 位の人に決めたいでしょ？　でも最初に出会った人がたまたま 1 位って確率はかなり小さいと思うの．だったら，もう少し様子を見たほうがいいんじゃないかなー」

「うん．それなら理に適った推論だ．正確に言えば最初の人が 1 位である確率は $1/n$ だ．逆に $1-1/n$ の確率で，その人は 1 位じゃない．では……，一体，何人くらい《様子見》すればいいのだろう？　これが考えるべき問題だ」

（うーん，何人くらい観察すればいいのかな？）青葉は目をつぶって考えた．

戦略あれこれ

「さっきの思考実験で，最初の何人かは見送ったほうがよさそうだって話をしたね」花京院が紙に長さが異なる棒を 10 本ほど書いた．人の姿を表しているらしい．

「うん，でも……，あんまり見送りすぎてもダメなんだよね？」

「ほどほどのところで，観察を終えないと，1 位を通り過ぎちゃうこともある．そうなったら手遅れだ」

「えーっと，$r-1$ 人まで観察して，r 人目に決めるってのはどう？　それで 1 位を選べる確率を最大化する $r-1$ を計算したらいいんじゃないかな？」

「いいアイデアだね．やってみよう」

「$r-1$ 人まで見送るってことは，残りが $n-(r-1)$ 人いるってことだから，r 人目が 1 位である確率は……，$1/(n-(r-1))$ じゃないかな？」

「いやいや，それなら $r-1=n-1$ 人見送ればいいことになってしまうから，おかしいんじゃないかな．もし $1/(n-(r-1))$ なら $r-1=n-1$ のとき

$$\frac{1}{n-(r-1)}=\frac{1}{n-(n-1)}=\frac{1}{n-n+1}=\frac{1}{1}=1$$

だからね」

「あ，ほんとだ．どこで間違ったんだろう？」

「君の計算だと $r-1$ 人見送ったときに，その中に 1 位がいないってことを暗に仮定している」

「あ，そうか．実際には最初の $r-1$ 人の中に 1 位が入っちゃう可能性だってあるんだね．うーん以外と難しいなあ」

「$r-1$ 人見送り，r 人目が 1 位である確率はこうだよ」

$$\begin{aligned}
&P(\text{見送った}r-1\text{人の中に1位がいない})P(r\text{番目に1位がいる})\\
&=\underbrace{\left(1-\frac{1}{n}\right)\left(1-\frac{1}{n-1}\right)\cdots\left(1-\frac{1}{n-(r-1-1)}\right)}_{P(\text{見送った}r-1\text{人の中に1位がいない})}\underbrace{\left(\frac{1}{n-(r-1)}\right)}_{P(r\text{番目に1位がいる})}\\
&=\left(\frac{n-1}{n}\right)\left(\frac{n-1-1}{n-1}\right)\cdots\left(\frac{n-(r-2)-1}{n-(r-2)}\right)\left(\frac{1}{n-(r-1)}\right)\\
&=\left(\frac{n-(r-1)}{n}\right)\left(\frac{1}{n-(r-1)}\right)=\frac{1}{n}
\end{aligned}$$

「あれえ??　$1/n$ になったじゃん．これって結局 $\cdots\cdots, r-1$ の値に依存しないから，当てずっぽうに 1 人選ぶのと変わらないってことだよね．うーん．ふりだしに戻ったかあ」

青葉は残念そうに言った．

$r-1$ 人見送り戦略

「$r-1$ 人目までは無条件で断るっていう部分はいいと思うんだ．問題はその後だね」花京院が続けた．

「そのあと？」

「せっかく $r-1$ 人分観察したのに，さっきのやりかたじゃ，その《経験》が生かされていない」

「うーん．そっかあ．どうしたら生かせるのかなあ」青葉が首をかしげた．

「今までに得た情報を活用すればいいんだよ．これまでに出会った人は記憶しているから，そのなかで一番よかった人のことは覚えてるはずだろ？　$\cdots\cdots$ いくら君でも」

「《いくら君でも》ってことはないでしょ．私，記憶力はいいほうなんだから」

「へえー $\cdots\cdots$，じゃあ，昨日のお昼，を何食べたか覚えてる？」

「ぐっ$\cdots\cdots$，もちろん覚えてるわよ．ただ女子としてここで公言することはひかえておくけど，で，その出会った人の記憶をどうすればいいの？」

「$r-1$ 人までの暫定 1 位を基準にして選べばいいんだよ．r 人以降に現れた人の中に，その暫定 1 位を超える人が現れたら，その人を選ぶんだ」

「なるほどー」

「$r-1$ 人目までは見送り，その中で暫定 1 位を決める．その後「暫定 1 位」を超える人が現れたらその人に決める．この方法を $r-1$ 人見送り戦略，略して《$r-1$ 戦略》と呼ぶことにしよう」

「ねえ，もし暫定 1 位を超える人が現れなかったらどうするの？」

「どういうケースかな？」花京院が楽しそうに聞いた．こういう場合，彼は確認のために質問していることが多い．そのことを青葉は知っていた．

「つまり，最初の $r-1$ 人の中に, たまたま 1 位が入ってたケースじゃないかな．その場合，r 人以降には，1 位がいないよね．暫定 1 位を決めるつもりが，最初の $r-1$ 人をパスしたときに，本当の 1 位を見逃しちゃった場合」

「その通り．$r-1$ 戦略でも失敗する可能性があるてことがポイントだ．だから，失敗する場合もふまえつつ，$r-1$ 戦略を使った場合に 1 位の人が見つかる確率を計算しなくてはならない」

「どうやるの？」

「まずはコンピュータに計算してもらおう」花京院は計算用のコードを書いた．

これで n 人分の順位をきめることができる」

```
runif(n);# n 人の得点ベクトル
```

10 人分のデータを作ってみよう．candidates（候補者）っていう変数に全員の情報を格納しておくよ．

```
candidates=runif(10)
# vector of 10 random variables.
```

```
# value of candidates are cashed for reference
candidates
```

```
##  [1] 0.90888802 0.99331394 0.07289098 0.75812803 0.70365264 0.95363874
##  [7] 0.77068887 0.44944402 0.30843969 0.15597654
```

これで，0〜1 の間の数値がランダムに 10 個生成され, その値は candidates という変数名に格納された．一番大きな数値が 1 位を表しているよ

「おー，ここまでは簡単だね」

「次に r-1 人まで観察した記録を保存する．見送られた人は refusedっていう変数に記録しておこう．これは次のように書けばいい」

```
refused=candidates[1:r-1];# 1〜r-1 人までの記録
```

r-1=4 の場合

```
refused=candidates[1:4];#r-1=4 人までの記録
refused
```

```
## [1] 0.90888802 0.99331394 0.07289098 0.75812803
```

確かに refused の中身は candidates の先頭の 4 人と一致している．

さて暫定 1 位を決めなくちゃいけない．これは no1_tempっていう変数にいれよう．

もう一度確認すると

candidates=runif(10);# n=10 人の得点ベクトル

refused=candidates[1:4];# r-1=4 人までの記録

だよ．

```
no1_temp=max(refused);#r-1=4 人までの暫定 1 位を no1_temp に記録
no1_temp
```

```
## [1] 0.9933139
```

refusedっていう変数は，refused=candidates[1:4]っていう意味だった．

max(refused) は max(candidates[1:4]) と書けばいいから，コードから省略しておこう．

```
no1_temp=max(candidates[1:4]);#r-1=4 人までの暫定 1 位を no1_temp に記録
```

次に暫定 1 位の no1_temp を使って，残りの n-r-1 人の中から暫定 1 位を超える人を探しだすコードだ．

「これはちょっとだけ難しいよ．やり方分かる？」

「えーっと，暫定 1 位の no1_temp よりも大きい値を candidates の 5〜10 番目から, 取り出せばいいんだね？」

「そうだよ．コードで書ける？」花京院が聞いた．

「私には難しいかな・・・・・・．花京院君，お願い」

「では最初はベタに for で if 文を回して，該当する要素を取り出すコードを書いてみよう」花京院はエディタにコードを打ち込んだ．

最初の 4 人は見送ったから，for を使って検索する範囲は for(i in 5:10){candidates[i]}だよ．

そのなかで no1_temp よりも大きい要素を取り出すから，取り出した要素を格納する変数を準備しておこう．

名前は選りすぐりの候補者だから selected とでもしとこうかな

```
selected=c();
for(i in 5:10){
  if(candidates[i] >no1_temp){
    selected=c(selected,candidates[i])}
}
selected
```

```
## NULL
```

どうかな？　うまくいったかな？

「うん，うまくいったみたいだね」青葉は注意深くコードを読み，for 文の前で selected=c() と代入した理由を質問した．

「それはね，あとで selected の最初の要素と，candidates の 1 位，つまり max(candidates が一致するかどうかを確かめるためだよ．仮に selected に誰も選ばれなければ第 1 要素は NULL だから 1 位 max(candidates とは等しくない．もし selected の第 1 要素である selected[1] が 1 位 max(candidates と同じだったら，r-1 戦略がうまくいったことを意味する」

「うーん，ムズカシーなあ」

「そういうときは簡単なベクトルを作って確かめてみるといいよ．ようするに

1.　探す範囲を決める 2.　取り出し条件を決める 3.　条件にあう要素を格納する変数を定義する 4.　検索

という流れができていれば，どんなコードでもいいからね．さて，実はこのコードには問題がある」

「問題？　ちゃんと動いてるみたいだけど」

「ちゃんと動くけど無駄が多い．selected の中には場合によっては 2 個以上の要素がはいることもある」

「そうだね．暫定 1 位の得点があまり大きくない場合は，それより点の高い人が，複数人いる場合があるよね」

「実際に必要なのは，暫定 1 位を上回る最初の一人だけだ．だから最初の一人が見つかった時点でループが止まるコードの方が効率がいい．そこで while を使う．条件を満たしている間だけ，指定したコードを実行する命令だよ，例えば　‘(1,5,2,6,8,10,12,5,6,4,8)’　の中で，最初に 7 を超える要素だけが欲しいとしよう」

```
a=c(1,5,2,6,8,10,12,5,6,4,8)
```

if 文を使うと無駄が多い

```
x=c();
for(i in 1:7){
if( a[i] > 7 ){ x=c(x, a[i])  }
```

```
}
x
```

```
## [1]  8 10 12
```

```
x[1]
```

```
## [1] 8
```

最初の 8 だけでいいのに，無駄な部分まで検索している．これでは一旦　x=(8,10,12,8)　というベクトルを作ってから，最初の要素をとりだなくてはいけない．一方，while を使えば，目当ての要素に辿り着いた瞬間にループは終わる．

```
x=c();
i=1;
while( a[i] < 7 ){ i=i+1; x=a[i] }
x
```

```
## [1] 8
```

ほら，a の中から 8 を見つけて，x に格納した瞬間に終わっている．

他にも方法はあるよ．例えば次のやり方は，いかにも Rっぽい．

```
a[a>7][1]
```

```
## [1] 8
```

「ええ？　これだけでいいの？」

「うん，R の特性を利用すれば，多分これが一番簡単なコードだと思う」

---

a>7 は a がベクトルのときは，論理値のベクトルを返すんだ．

```
b=a>7
b
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE
```

取り出し条件に b を代入すると, TRUE の要素だけが返ってくる．

```
a[b]
```

```
## [1]  8 10 12  8
```

最後にその最初の要素を取り出せば OK．

```
a[b][1]
```

```
## [1] 8
```

「うーん，なるほどー．知らなかったなあ」

「基本的には for と if と代入で，大体のコードはかけるもんなんだよ．でも言語の特徴を生かせば，より簡潔なコードがかけるんだ」

次の関数 sec0 は，n 人に対して r-1 戦略を適用する関数だよ．

```
sec0<-function(n,k){
  candidates=runif(n);# n 人の得点ベクトル
  no1_temp=max(candidates[1:k]);#k=r-1 人までのベストを記録
  selected=0; #r 人以降に no1_temp を超える人がいたら格納する変数. 初期値は 0
  s=k+1; #カウンタ用変数の定義．r=k+1 人目からサーチ
  while(candidates[s]<no1_temp &  s<n ) { #条件を満たす限りループ
    s=s+1;selected=candidates[s]} # candidates[s]>no1_temp でループ終了.
  # ここでカウンタ s は代入の前に進める．
  # selected には最初の candidates[s]>no1_temp を満たす candidates[s] が格納される
  print(c("no1_temp=","selected=","max(candidates)="))
  print(c(no1_temp,  selected,  max(candidates)))
  if(selected==max(candidates)){1}else{0} #selected が max(candidates) に一致するかどうか判定
};
```

n=100 にして，30 人まで無視して，31 人以降，ベストな人を選ぶ

```
sec0(n=100,k=30)
```

```
## [1] "no1_temp="      "selected="      "max(candidates)="
## [1] 0.9297754 0.9921718 0.9968526
```

```
## [1] 0
```

max(x) と x1 が一致していればサーチ成功

一致しない場合は，1 位を探せなかったという意味だ．

「プロトタイプ sec0 がうまくいったようなので，計算用に print を抑制した関数 sec を再定義するよ．中身は同じだ」花京院が新しいコードを書いた．

```
sec<-function(n,k){
  candidates=runif(n);# n 人の得点ベクトル
  no1_temp=max(candidates[1:k]);#k=r-1 人までのベストを記録
  selected=0; #k+1 人以降に no1_temp を超える人がいたら格納する変数. 初期値は 0
  s=k+1; #カウンタ用変数の定義．r=k+1 人目からサーチ
  while(candidates[s]<no1_temp &  s<n ) { #条件を満たす限りループ
    s=s+1;selected=candidates[s]} # candidates[s]>no1_temp でループ終了.
  # ここでカウンタ s は代入の前に進める．
  # selected には最初の candidates[s]>no1_temp を満たす candidates[s] が格納される
  if(selected==max(candidates)){1}else{0} #selected が max(candidates) に一致するかどうか判定
};
```

計算例

「n=100, r-1=37 の条件で 1000 回繰り返して成功の頻度を計算しみてよう．理論的には $1/2.718281828 \approx 0.3678794$ になるはずだよ」

```
mean(replicate(1000,sec(100,37)))
```

```
## [1] 0.351
```

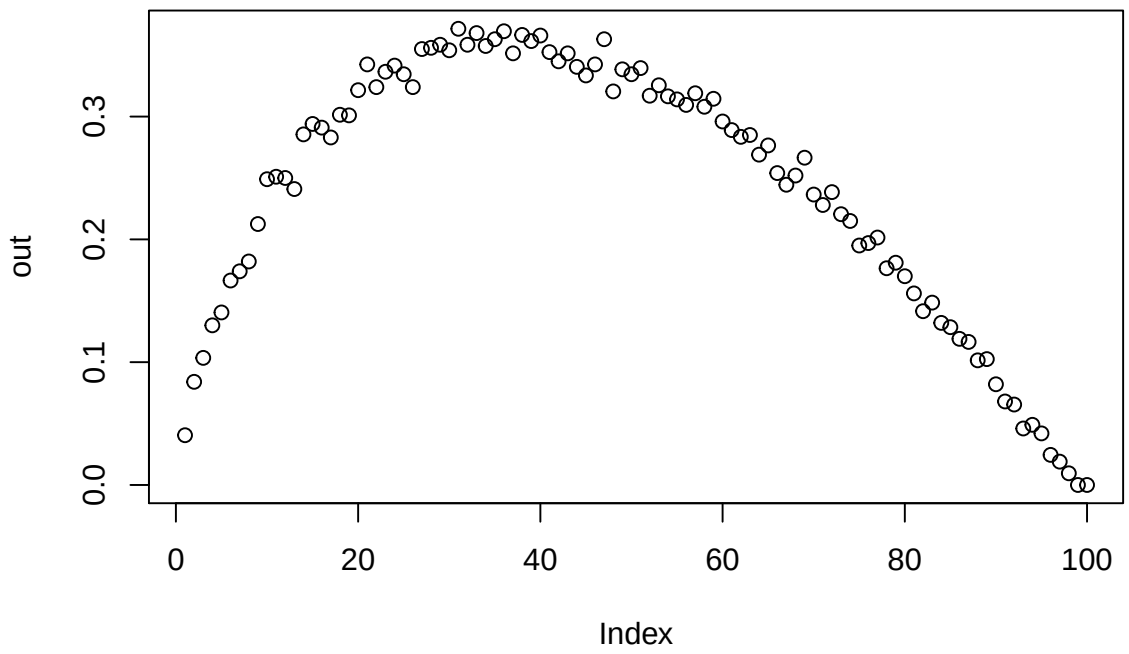
「r-1 の位置を変えながら，確率を比較してみよう」

```
test<-function(n,t,s0,s1){
  x <- c()   # 空の（要素数ゼロの）リストを作る
  for (i in s0:s1) {
    y <- mean(replicate(t,sec(n,i))) #
    x <- c(x,  y)   }#結果ベクトルに追加していく
  x}
```

「すこし時間がかかるよ．とりあえず 2000 × 100 で 20 万回ほど計算してみよう」

「そのあいだにコーヒーいれるね」青葉がコーヒーを入れる準備をした．

```
out<-test(n=100,t=2000,s0=1,s1=100)
plot(out)
```



「よーし，結果がでた．概ね r-1=36，37 あたりで確率が最大化している様子が分かるね」

「うーん，でもどうして r-1=36，37　あたりが一番大きいのかな？」

「たまたま，ってわけじゃないよ．なにせ, 1 つの固定した r-1 に対して，2000 回計算した平均値を比較してるからね」

理論

「うーん，どうして $r = 36, 37$ あたりで最大化するのかな？」青葉が聞いた．

「よーし最後の仕上げだ．$r-1$ 戦略を用いた場合に「1 位」に巡り会える確率を計算してみよう」

まず n 人の中で誰が一番かは事前に分からない．

そこで j 番目の人が一番であると仮定する．

$r-1$ 戦略を用いて，j 番目の人を選択できる条件とはなんだろうか？

$j \leq r-1$ のとき，1 位を見送ってしまうから，探し出せる確率はゼロだ．では $j > r-1$ のときどうか？

「それなら見送った $r-1$ 人の中には 1 位はいないから，うまくいくんじゃないかな？」

「ところがうまくいかない場合が存在する．例えば $r-1$ 人の中での暫定 1 位が，全体の中で 3 位だったと仮定する．すると r 人以降に，2 位の人に出会ってしまったら，その人を選ばなくてはいけない．その結果，1 位の人とは巡り会えない．だから，$r-1$ 戦略がうまくいくためには，j(1位のいる位置) の直前 $(j-1)$ 人の中で 1 位の人が，$r-1$ 人までに登場していないといけない」

「うーん，ちょっとややこしいな．もう少し分かりやすく説明してよ」

「じゃあ，こう考えたらどうだろう．$j-1$ 番目までの暫定 1 位が，見送った $r-1$ 人の中に含まれると仮定する．この暫定 1 位は，客観的には 3 位でも 4 位でもいい．ただ，$j-1$ 番目までに含まれる人々の中では 1 番良い人でなくてはならない．%この暫定 1 位は，別に 2 位でなくてもかまわない．3 位でも 4 位でもいいんだ．ただ，$j-1$ 番目までで，暫定 1 位でありさえすればい．すると，この暫定 1 位よりも r 番目以降で《いい人》は，必ず j 番目に現れる．そしてそれは，r 番目以降に現れた，暫定 1 位の人より《いい人》として，最初の人だから，必ず選ばれる．つまり客観的に 1 番のひとが選ばれるんだ．　だから $r-1$ 戦略がうまくいくためには，$j-1$ 番目までに含まれる人の中での暫定 1 位が，見送った $r-1$ 人の中に含まれることが必要なんだ」

「ちょっと図を書いてみないと分からないな」青葉は紙に絵を描きながら確認した．「ここが $r-1$ でここが j の位置，暫定 1 位がここにいると仮定すると……，あ，ほんとだ．ちゃんと 1 位の人が選ばれる」

「そうやって，図を描いたり，具体的な状況を想像するのはとてもいい方法だよ」

それでは，$r-1$ 戦略で 1 位の人に出会える確率を計算してみよう．さきほどの条件から

$j-1$ 番までの暫定 1 位が $r-1$ 番までに含まれ，かつ，j 番目に 1 位がいる確率ってことになる．

この確率は

$$P(j-1\text{番までの暫定1位が}r-1\text{番までに含まれる})P(j\text{番目に1位がいる}) = \frac{r-1}{j-1}\frac{1}{n}$$

j の位置は r 以降 n までありうるから，その全てを足し合わせると，$r-1$ 戦略が成功する確率となる．

$$\sum_{j=r}^{n} \frac{r-1}{j-1}\frac{1}{n} = \frac{r-1}{n}\sum_{j=r}^{n}\frac{1}{j-1} = \frac{r-1}{n}\sum_{j=r}^{n}\frac{n}{j-1}\frac{1}{n}.$$

ここで n が大きい場合の極限として

$$\lim_{n\to\infty}\frac{r}{n} = x$$

とおく．$t = j/n$ とけば, リーマン積分による近似によって

$$\lim_{n\to\infty} \frac{r-1}{n} \sum_{j=r}^{n} \frac{n}{j-1} \frac{1}{n} = x \int_x^1 \frac{1}{t} dt$$

である．この積分を解くと

$$x \int_x^1 \frac{1}{t} dt = -x \log x.$$

この確率を最大化する x は $x = 1/e \approx 0.367879\cdots$ である．

この結果からつぎのことが分かる．

最初の 36.8%は無条件でパスして，その中にいる暫定 1 位を覚えておく．それ以降最初に暫定 1 位を超える人が現れたらその人を選ぶことで，集団の中で 1 位の人と巡り会う確率を最大化できる．

「おー，ちゃんと計算するとちゃんと 36.8%っていう数値が出てくるんだね．なんだか不思議ー」

「仮定の中にはなにも具体的な数字が出てこないのに，確率を最大化させるアルゴリズムに基づいて計算すると，具体的な数字がでてくるのがおもしろいところだね」花京院は，計算結果を確かめるために，紙とペンをとった．

（36.8%かあ $\cdots\cdots$，私はこれまでに何$\%$の人と出会ったのかなあ）

計算に再び熱中する花京院を眺めながら，青葉はコーヒーカップに唇をあてた．

References

Ferguson, Thomas, 1989, “Who Solved the Secretary Problem?” *Statistical Science*, 4(3): 282-296.