運命の人と出会う確率

花京院と青葉の文体練習 3 *

浜田　宏

最大の効用や満足を獲得しようと努力する個人はまた,《理性的に》行動するといわれている．しかしいまのところ，理性的行動の問題を満足できる形で扱ったものはないといっても過言ではない (Neumann and Morgenstern 1953=2009:47).

登場人物 [1]

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.

1　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一人静かにシミュレーション用のコードを書いていた花京院は，ふうっとため息つくと，青葉の方に顔を向けた.

「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．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」

---

*ver 1.0 (2016 年 6 月 09 日), ver 1.1 (2016 年 6 月 11 日), ver 1.2 (2016 年 8 月 25 日)

[1]注) この二人は『数理社会学入門—ベルヌーイ変奏曲』に登場する人物です.
(http://www.sal.tohoku.ac.jp/~hamada/にて公開中)

「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．その問題は 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) って呼ばれることが多いけど，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，という文脈で考えてみよう」

2　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

2.1　仮定の意味

「n 人のなかから一番いい人を選ぶ方法は何だと思う？」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』ってやり方が一番簡単じゃないかな」

「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

「まあ，仮定だからと言ってしまえばそれまでなんだけど．1 つの解釈としては《競争の激しさ》を表していると考えられる．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」

「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．例えば不動産を見て回る状況なんかが似ているよ」

「不動産？」

「アパートを借りるとき，早く決めないと別の人に借りられますよって言われたことない？」

「あー，あるある．不動産屋さんに『今決めないと，他の人が契約しちゃうかもしれませんよ』って言われたことあるよ．あれ言われると，あせるんだよねー」

「恋愛市場も同じで，早く決めないと誰かにとられちゃうんだ．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》を探し出す問題と考えることもできる．もう少し，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」青葉が身をのりだした．

2.2 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に理由を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じゃん．だったら，もう少し様子を見たほうがいいかなって思ったの」

「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．

2.3　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」

$$

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

$$

「あれえ??　$1/n$ になっちゃったじゃん．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．ふりだしに戻っちゃったね」青葉は残念そうに言った．

3　$r-1$ 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っ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[2].

$$

次のコードでn人分の乱数を発生させて，ランダムな順位をきめる．

```
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); # 10 人の得点ベクトル
```

```
##  [1] 0.82390068 0.68132518 0.62727770 0.39724300 0.02550097 0.05921819
##  [7] 0.70666825 0.44717369 0.41982160 0.98796952
```

0〜1の間の数値がランダムに10個生成され, その値がcandidatesという変数名に格納されたよ．一番大きな数値が1位を表しているよ．この場合，たまたま最後の人（0.98796952）が1位だね．アウトプットの [1] とか [7] は，その右横の数字がベクトルの何番目の要素であるかを表している．[7] 0.70666825は，7番目が0.70666825っていう意味だよ．

$$

[2]以下，本文に登場するコードはRを用いた．コピペしてRで計算することができる．また文体練習3.1 (http://rpubs.com/HHamada/193736) はRから直接出力したhtmlである

「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」

$$

$1 \sim r-1$ 人までの記録は, refused=candidates[1:r-1] と書けばいい．もし r-1=4 の場合は

```
refused=candidates[1:4];#  r-1=4 人までの記録
## [1] 0.8239007 0.6813252 0.6272777 0.3972430
```

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

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．さて， refused の中から最大値を選んで，それを no1_temp という変数に代入しよう．

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

ここで，max(refused) は max(candidates[1:4]) と書いてもいいから，コードから省略しておこう．すると，

```
no1_temp=max(candidates[1:4]);#r-1=4 人までの暫定 1 位を no1_temp に記録
## [1] 0.8239007
```

$$

「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？」花京院が聞いた．

「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？ うーん，ちょっと私には難しいかな……．花京院君，お願い」

「ではオーソドックスに 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追加の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．

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
## [1] 0.9879695
```

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位とは等しくない．もしselectedの第1要素であるselected[1]が1位と同じだったら，$r-1$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んだよ．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．while は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 (5,2,8,10,6) 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(5,2,8,10,6)
x=c();
for(i in 1:5){
if( a[i] > 7 ){ x=c(x, a[i])  }}
x[1]
## 8
```

こんな風に for 文を使って書ける．

でも if 文を使うとちょっと無駄な部分がある．条件に一致する 8 だけ取り出して終わればいいのに，最後まで検索している．

いま書いたコードだと，x=(8,10)　というベクトルをいったん作ってから，x[1] の部分で最初の要素をとりだしているんだ．

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．

こんな感じだよ

```
a=c(5,2,8,10,6);
x=c();
i=1;
while( a[i] < 5 ){ i=i+1; x=a[i] }
x
## 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=c(5,2,8,10,6);
a[a>7][1]
## 8
```

$$

「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．

例えば a の中身が a=c(5,2,8,10,6) だとしたら，a>7 は

(FALSE, FALSE, TRUE, TRUE, FALSE)

だよ．これを条件指定に使えば， TRUE の部分だけ値が返ってくる．

数学風に書けば，こんなイメージかな

```
a[a>7]=a[c(FALSE, FALSE, TRUE, TRUE, FALSE )]=c(8, 10)
```

$$

「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,r-1){
  candidates=runif(n);# n 人の得点ベクトル
  no1_temp=max(candidates[1:r-1]);#r-1 人までのベストを記録
  selected=0; #r-1+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=r-1+1; #カウンタ用変数の定義．r-1+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
};
```

$n = 100$ と仮定しよう．30 人まで無視して，31 人以降ベストな人を選ぶ戦略は，こうなる．

```
sec0(n=100,r-1=30)
```

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.

```
# [1] "no1_temp="         "selected="         "max(candidates)="
# [1] 0.9513356             0.9780149          0.9780149
# [1] 1
```

$$

「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,r-1){
  candidates=runif(n);# n 人の得点ベクトル
  no1_temp=max(candidates[1:r-1]);#r-1 人までのベストを記録
  selected=0; #r-1+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=r-1+1; #カウンタ用変数の定義．r-1+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
};
```

$$

4 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)))
```

「$r-1$ 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 $r = 36, 37$ あたりで確率が最大化している様子が分かるね．これは，たまたま，こうなったけじゃないよ．なにせ，　1 つの固定した r に対して，2000 回計算した平均値を比較してるからね」

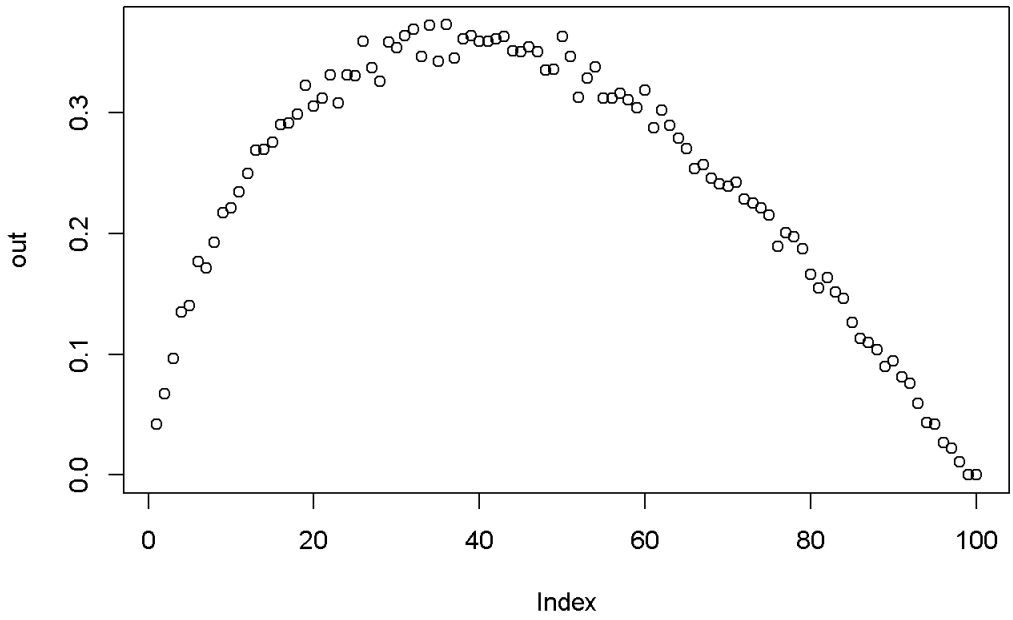

図 1: シミュレーションの結果

5　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．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……，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.