社会科学のための Mathematica と R 入門

浜田宏

1. イントロダクション

1.1. Mathematica を使う理由

数理社会学のモデルは，基本的には単純で解析可能なモデルを目指すべきである．シミュレーションは，あくまで解析不可能なモデルを分析するための次善の方法と考えるべきだ．ゆえにシミュレーションを作る場合も，不必要に複雑なものではなく，なるべく単純なモデルを目指すべきである．その目的に適したソフトが Mathematica であり，以下の三つの特徴を備えている．

- 直接的に数学ができる
  - 名前から分かるとおり Mathematica は数学に特化した言語であり，様々な数学計算が可能である．シミュレーションだけでなく，数理社会学の中心たる解析的モデルの作成・分析にも役立つ．例えば論文内で使う命題を証明するための予想計算や，手計算の結果確認に Mathematica を利用できる．
- プログラムの作成時間が短い（コードそのものが短い）
  - 研究者にとって必要なのはアプリケーションではなく，特定の条件下での計算結果である．したがって計算時間を短縮させることより，プログラムの作成時間を短縮させる方が，圧倒的に効率的である．
- グラフィック出力とインタラクティブなインターフェイスの作成が簡単
  - 関数のグラフを簡単に作成できる．また Manipulate によってインタラクティブなユーザーインターフェースが簡単に作成できる．
  - Manipulate から生成したアニメーションをパワーポイントに埋め込むことで，動的なプレゼンテーションを作成できる．

1.2. R を使う理由

データ分析をする場合に，reproducieble なコードを書いてから実行すると，便利である．データの読み込み，リコード，変数の変換，分析，結果のアウトプットまでの一連の流れを R のようなプログラム言語を使って記述すると，ワンボタンで再現できる．使いやすさ・安定性・正確さ・拡張性・導入コストの点から考えて，しばらく R は研究者にとって標準的な言語になるだろう．また階層ベイズ MCMC のような，モンテカルロシミュレーションを含む分析を実行する場合でも, R は標準的なソフトである(現状では BUGS 系ソフトや STAN の力を借りているが，やがて統合されるだろう).

以下簡単に特徴を確認する．

- 豊富な CRAN パッケージ
- 関数型プログラム，リスト型演算によるコードの簡略化
- オライリーが詳細なマニュアル本を発行

無料（学生にとっては最大の誘因）

1.3. これから始める人のための 10 大原則

1. Mathematica に特有の入力手法（ノートブック）を覚える
   - 「セル」という入力・出力の単位を理解する
   - 新たな命令は空白セルを一番下に追加してから書く
   - セルの階層性を理解して展開・格納を活用する．「alt+4」や「alt+5」で指定したセルに階層性を持った見出しを設定できる
   - 階層的なセクションを活用するとコードの全体像が理解しやすい
   - 入力は基本的に半角英数文字だけを使う．全角文字はセクションタイトルとコメント内だけで使う
   - セミコロン(;)の入力忘れに注意する. 逆に余計な場所にセミコロンを入力すると，結果が抑制されるので注意する
   - 括弧の種類に注意する． ( )と[ ]と{ }はそれぞれ意味が異なる．

2. F1 でヘルプがいつでも呼び出せる．
   - 内蔵関数は選択してから F1 キーを押すと，直接定義にジャンプできる

- ヘルプ上の例は全て直接編集できる（コピー&ペーストできるので例をコピーしてパタメタだけを変更するとよい）
- Mathematica ほどヘルプが充実しているソフトは，なかなか存在しない

3. コメント (* *) は他人にも分かるように書く
   - ここでいう他人は「未来の自分」も含む．自分が書いたコードでも，時間がたてば内容を忘れてしまう．誰が見てもコードの意味が分かるようなコメントが望ましい
   - またコードそれ自体もなるべくシンプルに書くことで，可読性を高めるべきである

4. サンプルコードを編集するときは必ずコピーする
   - プログラミングは試行錯誤の繰り返しである．作業が逆行しないように，新たな作業を始める場合は，今までの作業を保存しよう．
   - 慣れるまでは，一行ずつコードを追加するつもりで進めよう．一行ずつ追加すれば，どこでエラーが生じたかをすぐに確認できる（実験の統制と同じだ）．
   - 問題なく動いている関数でも，修正することでエラーが出てしまうことがある．少なくとも修正前に復帰できるように，必ずコピーをとってから修正作業に入る
   - 大きな変更があった場合は，ノートブック全体をリネームしてコピーする
   - 一つのノートブック内に複数の見出しを設定して全体の構造が一目で把握できるように整理する

5. 最初に単純なプロトタイプを作る
   - 「千里の道も一歩から」ともいう
   - いきなりサイズの大きなパラメタを指定しない．エージェントの数は最初は 5〜6 人で十分であり，反復回数も最小の 2 回でよい
   - プロトタイプを自力で作るのは難しい．文法や語彙を修得するまではウルフラム・デモンストレーション・プロジェクトの中から，自分の作りたいプログラムのプロトタイプを探す．http://demonstrations.wolfram.com/
   - 自分で作りたいものが作れるようになると，プログラムは劇的に楽しくなる．原理的には，デジタル上ならばどんな世界でも作ることができる（ただし，あまり深入りすると，あちらの世界から戻れなくなるので注意しよう）．

6. グローバルな定義とローカルな定義を使い分ける
   - グローバルに定義した変数は，別のノートブックでも維持されるので注意する

- コードが短い場合はグローバル定義を使っても良いが，ある程度長くなってきたら Module 内に入れて，ローカル変数として使う

7. Ctrl+Shift+B で括弧の対応を確認する．
   - 括弧の階層が深くなると，対応が分からない．そんなときは，括弧の最初か最後にカーソルを合わせて，Ctrl+Shift+B を押す．
   - 新しいバージョンでは変数や括弧の対応をある程度色で教えてくれるので参考にしよう

8. デバッグには Print を活用せよ
   - Module は原則，最後の結果しかリターンしない．途中経過を知りたい場合は，変数を Print で強制出力する．
   - 複数の `Print` を使う場合は，`Print[“x1= ”, x1]; Print[“x2= ”, x2]` などとラベルを使って区別すると分かりやすい．
   - エラーが生じた箇所はエラーメッセージから特定せよ．
   - エラーが生じた箇所が分からない場合は最初の一行から順番に `Print` で強制出力せよ．

9. 変数の名前の付け方を工夫せよ
   - 名前を見ただけで，意味が分かるような変数名が望ましい．
   - 好みの問題はあるが，`AgentPosition` とか `StrategyProfiles` など読めば中身が分かるキャメル・ノーテーションを利用する．
   - 例えばグローバル変数は全て大文字，ローカル変数やユーザー定義関数は小文字，などの一貫したルールを用いる．
   - 変数名をリネームする場合は，必ず「置換」機能を使う．目視で探してはならない（見逃した変数があると，それが原因でエラーが起こる）．選択した範囲をまとめて置換すれば，ミスがない．

10. シミュレーションや数理モデル分析を成功させるために，もっとも重要なことはアイデアである．
    - 基本的なアイデアやデザインさえよければ，技術や経験が足りなくても良いプログラムやモデルを作ることが可能である．アイデアを実現するために技術があるのであり，その逆ではない．

- 複雑なアルゴリズムは，事前に紙に簡単な構造を書くと，コード化しやすい．時には画面から離れて紙の上で考えよう．逆に簡単な手続きは，わざわざチャートを書く必要はない．
- 「精神なき専門人」になってはいけません（ヴェーバーの言葉）
- 「頭を悪くしてはいけません．根気づくでおいでなさい」（漱石の言葉）

入力支援に関する注意

Ver6 (7?) 以降，Mathematica には入力支援機能が追加され，関数名を途中まで書けば候補を予想してくれるようになった．基本的には便利は機能だが，注意すべき点がある．それは一時的代入を意味する x->2 の矢印部分を自動的に半角フォントの→に変換してしまうことである．サンプルプログラム内の矢印を全角矢印と勘違いして，代入のつもりで全角→を入力すると当然エラーになる．コードは絶対に半角文字で入力すること．

2．教育達成の出身階層間格差　edu.nb

次のような状況をプログラムで表現してみよう．

社会に上層と下層の二つのグループが存在する．初期条件として子供の成績は正規分布にしたがう．上層出身の子供は，下層出身の子供より平均値が高い．ただし分散は同じ．
全体の上位 x 人だけが進学できる．教育機会 x の増加と共に，出身階層間の進学格差はどのように変化するか？

コード例

```
Module[{x, score1, score2, overall, n = 20},
 x = 20;
 score1 = Table[{RandomReal[NormalDistribution[52, 1]], 1}, {n}];
 score2 = Table[{RandomReal[NormalDistribution[50, 1]], 0}, {n}];
 Print["score1=", score1];
 Print["score2=", score2];
 overall = Flatten[{score1, score2}, 1];
 overall = Sort[overall, #1[[1]] > #2[[1]] &];
 Table[overall[[i]][[2]], {i, 1, x}]
 ]
```

解説.
最初に Module 内のローカル変数に，x, score1, score2, overall, n を用意した．ローカル変数は，あらかじめ決めるより，プログラムを書きつつ必要になったらその都度加えていくほうが自然である．それぞれの意味は次の通りである．

x,　**社会全体で進学可能な人数**
score1,　**上層出身者の成績リスト. 成績と出身階層の順序対**
score2,　**下層出身者の成績リスト. 成績と出身階層の順序対**
overall, **二つの集団を合わせたリスト（l エルと 1 いち, は混同しないように注意する）**
n　　**各集団の人数（共通）**

n=20 だけは初期値が与えられている．これは本文に書いてもよい．社会全体では上層 20+下層 20＝40 人が存在する．最初の一行「x = 20;」は進学可能人数が 20 人であることを意味している（つまりこの社会の進学率は 50％である）．次の score1 と score2 には，正規分布にしたがう成績が，それぞれの人数分格納される．

```
Table[{RandomReal[NormalDistribution[52, 1]], 1}, {n}]
```

は平均 52，標準偏差 1 の正規分布に従う乱数を n 個生成せよ，という命令である．この場合 n=20 と指定したので 20 人分の成績が生成され，score1 という変数に格納される．ここで注意すべきは，後で出身階層の情報を参照するために，

{成績，1}，{成績，1}，　...　，{成績，1}

という数値のペアをリストにしていることである（これを順序対という）．右側の数字 1 は「上層」を表すコードである．

三行目も基本的に同じだが，平均が上層に比べて小さく（これは下層出身者の方が平均値が低い，という仮定に基づいている），また出身を示すコードには「0」が割り当てられている．

{成績，0}，{成績，0}，　...　，{成績，0}

```
Print["score1=", score1];
Print["score2=", score2];
```

は，成績リストの具体的な数値を見るために変数 score1 と score2 をプリントしている．こうすることでプログラム途中の変数に，どのような値が格納されているかを確認できる．

```
overall = Flatten[{score1, score2}, 1];
```

は二つの成績リストを一つにまとめている．Flatten という命令は，リストの階層を調整するために使っている．

「上位の x 人が進学できる」という仮定は，正確に言えば「上層も下層も一緒に試験を受けて，出身階層に関係なく，成績が高い上位の x 人が進学できる」という意味である．したがって，リスト overall を成績順にソートして上位 x 人を選ぶ必要がある．

```
overall = Sort[overall, #1[[1]] > #2[[1]] &];
```

は成績に基づいてソート（並び替え）している．ただしリストは出身階層コードを含むので，それを無視して，成績のみに注目してソートするよう工夫してある（純関数を使ってリストの一番目の要素だけを比べている）．

```
Table[overall[[i]][[2]], {i, 1, x}]
```

最後に成績順に並べたリストから上位 x 人だけを抽出している．

2.1. エスケープ入力

リストの位置を示す二重括弧は，見づらく間違えやすい．そこで「エスケープ入力」を使う．

```
Esc, [, [, Esc
```

`[[` は自動的に 1 文字の白抜きブラケット〚 に変化する．閉じる括弧は

```
Esc, ], ], Esc
```

である．括弧の対応は Ctrl+Shift+B で確認できる（2 回目）．

2.2. 結果表示の工夫

Q. 上記のプログラムからどのような興味深い分析結果が導けるかを考えよ．どんな index に着目して，どんなパラメータを変化させればよいだろうか？ また分析結果をオーディエンスに分かりやすく伝えるために，どのような結果の可視化が考えられるだろうか？
考えてみよう．

解答例．相対リスク比あるいはオッズ比を使って上層と下層の進学率格差を指数化する．この指数が x の変化に対して，どのような変化するのかをグラフで表示する．

2.3. オッズ比の定義

例（医者の再生産）．親と子の職業に関して次のようなデータが得られたとしよう．このクロス表から「親が医者の人は，そうでない人にくらべて，医者になりやすい」といえるだろうか？

		子			
		医者		医者以外	
親	医者	10	①	5	②
	医者以外	20	③	40	④

親が医者の場合，その子供の「医者と医者以外の比率」は

$$\text{医者/医者以外} = 10 / 5 = 2$$

である．

一方，親が医者以外の場合，その子供の「医者と医者以外の比率」は

$$\text{医者／医者以外} = 20 / 40 = 1 / 2 = 0.5$$

である．出身によって「医者へのなりやすさ」がどの程度違うのかを比較するために両者の比をとると

$$\frac{①/②}{③/④} = \frac{10/5}{20/40} = \frac{2}{0.5} = 4$$

となる．つまり，「親が医者である人は，親が医者でない人に比べて，4 倍医者になりやすい」といえる．この 4 倍という数値を「オッズ比」という．

一般に「ある条件下での，注目するアウトカムの生じやすさ」を示すオッズ比は，

		結果が発生	発生しない
父	条件あり	a	b
	条件無し	c	d

というクロス表にもとづき，定義される．

$$\text{結果のオッズ比} = \frac{a/b}{c/d} = \frac{ad}{bc}$$

ちなみにリスク比（相対リスク比）の定義は

$$\text{結果のリスク比} = \frac{a/(a+b)}{c/(c+d)}$$

である．

Q　edu.nb を参考にして，つぎの状況をコード化せよ

50 人の生徒に，正規分布に従う学力と親の収入が与えられる．親の収入が上位 50%以内の生徒だけ進学できる仮定した場合，進学できた子供のリストを表示せよ．

ヒント：学力と収入の順序対を要素とするベクトルをつくる．次に順序対の収入のメディアンを計算し，条件文に使う．最後に条件を満たす順序対のうち，学力だけを表示する

Q　上記のモデルにおける不平等の指数として，オッズ比を定義して計算せよ．

2.4. 喫煙と肺ガンの関係

喫煙者と非喫煙者の肺ガン発生を調査したところ，以下のようなデータが得られた．

	発ガン	発ガンしない
喫煙する人	10	40
喫煙しない人	50	400

Q　オッズ比を計算せよ．

計算の結果「煙草を吸う人は吸わない人に比べて***倍，肺ガンになりすい」ということがわかる．

なお厚生労働省は以下の情報をウェブサイトで掲示している（2015 年 10 月現在）．

- 喫煙男性は、非喫煙者に比べて肺がんによる死亡率が約４．５倍高くなっているほか、それ以外の多くのがんについても、喫煙による危険性が増大することが報告されています。
- 喫煙は世界保健機構（ＷＨＯ）の国際がん研究機関（IARC）において発がん評価分類でグループ１（人間に対して発がん性あり。人間に対する発がん性に関して十分な証拠がある）に分類されています。
- 喫煙により空気の通り道である気道や肺自体へ影響を及ぼすことが知られています。このため、喫煙は慢性気管支炎、呼吸困難や運動時の息切れなどの症状が特徴的な肺気腫や喘息等の呼吸器疾患の原因と関連しています。さらに歯周病の原因と関連があるという報告があります。
- 喫煙者は、非喫煙者に比べて虚血性心疾患（心筋梗塞や狭心症等）の死亡の危険性が１．７倍高くなるという報告があります。また、脳卒中についても喫煙者は、非喫煙者に比べて危険性が１．７倍高くなるという報告があります。

喫煙者は「煙草を吸ったからといって必ずガンになるわけじゃない」と，よくいう．これは論理的には間違いではないが「煙草を吸うとガンになるリスクが増加する」ことも事実である．喫煙は健康だけでなく美容にも悪い．また本人だけでなく胎児にも悪い．禁煙にはニコレットがお薦めである．

2.5. R コード

edu.r

#関数 edu2 が計算用，rr2 は進学者数の変化に伴う結果を plot する関数

```
edu2<-function(x,n1,n0){
  #x:進学者数     n1:上層人数       n0:下層人数
  df1 <- data.frame(score = rnorm(n1, mean = 52, sd = 1),id = rep(1,n1))
  df0 <- data.frame(score = rnorm(n0, mean = 50, sd = 1),id = rep(0,n0))
  #df1,上層 n1 人の成績データ       df0:下層 n0 人の成績データ
  #score,id は変数名ラベル．結合のため同じ名前を使う
  df <- rbind(df1, df0)
  #データフレーム df へと，df1 と df0 を行結合して格納する
  #同じ列名でないとマッチしないので注意する
  df<-df[order(df$score, decreasing = TRUE), ]  #成績降順で並び替え
  x1=head(df$id, n=x) #上位 x 人の id 抽出
  rr <- (sum(x1)/x)/(1-sum(x1)/x)#相対リスク比.上層進学率/下層進学率
  return(rr)}

rr2<-function(n1,n0){
  #n1:上層人数       n0:下層人数
  end<-n1+n0
  result<-Inf#結果の格納．最初は下層進学者 0 人のため，inf を入れておく
  for(i in 1:end){
  result<- c(result,edu2(i,n1,n0))}#結果を for 文で result に追加
  #ここで先に定義した関数 edu2 を使う．進学者数を i で変化させる
    xop<-0:end #plot 用に x 軸を定義
  plot(xop,result)
}

rr2(1000,1000)#計算実行例．上層人数，下層人数の順に入力
```

2.6. 厳選　特によく使うコマンド一覧（アルファベット順．一言解説付き）

Abs	絶対値をとる
All	行列の列指定に使用
And (&&)	論理演算のアンド
Append	リストに要素を加える
AppendTo	リストに要素を加え，更新する
Apply	Apply[Plus, {a, b, c, d}]でリストの総和
ArrayPlot	配列を可視化する
Axes	グラフの軸
AxesLabel	軸のラベル
AxesOrigin	グラフの原点
BarChart	棒グラフ
Binomial	二項係数
Cases	指定したケースを選択
Clear	定義のクリア
Collect	ある変数で多項式を整理する
Compile	関数を高速化する
Complement	補集合をとる
ContourPlot	等高線プロット
Count	条件に合致する要素を数える
D	微分
Delete	指定位置の要素を削除
DeleteCases	条件に合致する要素を削除
Directory (SetDirectory)	現行ディレクトリを返す．データの記録などに使用する
Do	反復．For の簡易バージョン
Dot	ドット積
Drop	指定位置の要素を削除
DSolve	微分方程式の解
E	自然対数の底（無理数）
Eigenvalues	固有値の計算
Eigenvectors	固有ベクトルの計算
Equal (= =)	論理演算用の条件．代入の＝と区別する

Erf	オイラー関数
Exp	指数関数
Expand	代数式の展開
Factor	多項式の因数分解
Factorial (!)	階乗
FindRoot	方程式の数値解
First	リストの最初の要素を取得
FixedPoint	不動点
Flatten	リストの平滑化
Floor	最も近い整数に切り下げる
FontSize	フォントサイズの指定
For	ループ
Gamma	ガンマ関数
Graphics	グラフィック出力
GraphicsArray	グラフィックの配列
Greater (>),GreaterEqual(>=)	不等号＞
Hue	色の指定
If	条件を満たす場合に命令を実行する
Increment (++)	変数に１追加する
Infinity	無限大
Input	データのインプット
Integer	整数の型指定
Integrate	積分
Intersection	共通集合
Join	リストの結合
Last	リストの最後をとりだす
Length	リストの長さを取得する
Less (<), LessEqual (<=)	不等号＜
Limit	極限値をとる
LinearPrograming	線形計画法
ListPlot	リストのプロット（離散値に使う）
Log	対数
Map	関数の適応

MatrixForm	行列形式で表示する
MatrixPower	行列のべき乗
Max	最大値
MemberQ	指定要素を含むかどうかのチェック
Min	最小値
Module	局所変数を含む一連の手続きのまとまり
N	近似
NDSolve	近似解
Nest	関数をネストして適用する
Not	条件子　否定
Or	条件子　あるいは
Order	順番に並べる
Part ([[]])	リストから指定した位置の要素を取得
Pi	円周率
Plot	連続関数のプロット
Plot3D	二変数間数の三次元プロット
PlotJoined	離散プロットをつなぐ
PlotRange	プロットする範囲の指定
PlotStyle	プロットした線の色・太さなど
Position	リストから指定要素の位置を探す
Power (^)	べき乗
Prepend	リストの最後に要素を追加する
PrependTo	リストの最後に要素を追加して更新する
Prime	素数を取得
Print	強制的に出力する
Product	かけ算をまとめて実行
Random	乱数
RandomInteger	乱数整数
RandomReal	乱数実数
Range	1 から x までのリスト
Remove	関数を消去
ReplacePart	置換
Rest	リストのある位置から後ろの部分

Reverse	逆順に並べる
RGBColor	色指定
Round	四捨五入
Rule (->)	パタンの一時的代入
SameQ	条件子
Select	条件を満たす要素を選択
SeedRandom	乱数のシード
Sequence	関数に自動的に与えられる引数の列を表す
Series	テイラー展開
SetDirectory	ホームディレクトリの指定（データ出力に使う）
Show	グラフィックスの表示
Solve	方程式を解く
Sort	並び替え
Sqrt	平方根
Sum	総和
Table	ルールに従ってリストを作成
TableForm	表形式で出力
Take	リストの要素抽出
Thick	ラインや文字を太く
Ticks	目盛り
Timing	計算時間
Total	リスト要素の総和
Transpose	行列（リスト）の転置
Union	和集合
Which	条件分岐．三パタン以上の分岐に便利
While	条件を満たすまで反復

これだけは覚えるべき最低限の関数（神 7）

Table, If, For, Plot, Random, Part([[]]),Select

3．Module からの関数化　edu.nb

Module で一連の計算手続きを定義した後，異なる条件で計算結果を比べる場合には，関数化すると便利である．例えば教育達成のモデルで，全体進学者数 x の増加の効果を知りたい場合，次のようにコードを修正する．

```
Module[{x, score1, score2, overall, n = 20},
 x = 20;
 score1 = Table[{RandomReal[NormalDistribution[52, 1]], 1}, {n}];
 score2 = Table[{RandomReal[NormalDistribution[50, 1]], 0}, {n}];
overall = Flatten[{score1, score2}, 1];
 overall = Sort[overall, #1[[1]] > #2[[1]] &];
 Table[overall[[i]][[2]], {i, 1, x}]
 ]
```

関数化：

Module **から引数に指定する変数** x **を消去して，冒頭で** edu[x_]:= **と宣言する．**

edu[ ]**は関数名なので自分の好きな名前をつけるとよい．**

```
edu[x_]:= Module[{score1, score2, overall, n = 20},
 score1 = Table[{RandomReal[NormalDistribution[52, 1]], 1}, {n}];
 score2 = Table[{RandomReal[NormalDistribution[50, 1]], 0}, {n}];
  overall = Flatten[{score1, score2}, 1];
 overall = Sort[overall, #1[[1]] > #2[[1]] &];
 Table[overall[[i]][[2]], {i, 1, x}]
 ];
```

新しく定義された関数 edu[ ] は x を引数として，Module 内で指定された手続きを実行する．例えば x=5 の場合の計算結果がほしいときは

```
edu[5]
```

と入力する．x=10,x=15 の場合の計算結果がほしいときは

```
edu[10]
edu[15]
```

と入力するだけでよい．

複数の引数を指定する場合には

```
edu[x_,n_]:= Module[{score1, score2, overall},
 score1 = Table[{RandomReal[NormalDistribution[52, 1]], 1}, {n}];
 score2 = Table[{RandomReal[NormalDistribution[50, 1]], 0}, {n}];
  overall = Flatten[{score1, score2}, 1];
 overall = Sort[overall, #1[[1]] > #2[[1]] &];
 Table[overall[[i]][[2]], {i, 1, x}]
 ];
```

のように，引数を並べて宣言する．このとき順番が重要で，x=10,n=30 という条件で計算したい場合は

```
edu[10, 30]
```

と指定する．

Q　成績の平均値を出身階層別に関数の引数にしなさい．

解答例．正規分布の平均を引数にするには，次のように関数を定義する．

```
edu[x_,n_,m1_,m2_]:= Module[{score1, score2, overall},
 score1 = Table[{RandomReal[NormalDistribution[m1, 1]], 1}, {n}];
 score2 = Table[{RandomReal[NormalDistribution[m2, 1]], 0}, {n}];
  overall = Flatten[{score1, score2}, 1];
 overall = Sort[overall, #1[[1]] > #2[[1]] &];
 Table[overall[[i]][[2]], {i, 1, x}]
 ];
```

ユーザーが定義した関数は，他の関数内で自由に使うことができる．例えば，x の変化を知りたければ，

```
  Table[ edu[x,20,52,50], {x,5,40}]
```

のように Table 内で呼び出せばよい．

4. グローバル企業の憂鬱

グローバル人材の育成は，ある条件下では生産性の低下をもたらすことを単純な数値計算で示す．企業が採用する人材には《語学力》と《生産力》の二つの能力が存在して，この二つは 2 次元正規分布に従うと仮定する．企業が語学力を基準に上位 $x\%$ を採用する．すると《語学力》と《生産力》が正の相関を持つと仮定しても，《生産力》基準で上位 $x\%$ を採用した場合に比べて，平均的な生産能力は必ず低下する（浜田 2013）．

```
f001[r_(* 能力と語学力の相関 *)] :=
 Module[{score1, score2, all, a1, a2, new1, new2, n = 200, x = 10,  m1 = 50(*能
力平均*),  m2 = 50(*語学力平均*), s1 = 5, s2 = 5},
  (*学生は仕事能力と語学力の二つを能力として持つ*)
  all = RandomReal[ MultinormalDistribution[{m1, m2}, {{s1^2, r*s1*s2},
 {r*s1*s2, s2^2}}], {n}];  (* 乱数の発生  ver7 では RandomVariate  を使えない*)
  a1 = Sort[all, #1[[1]] > #2[[1]] &];(* 能力によるソート Print["a1= ",  a1];*)
    new1 = Table[a1[[i]][[1]], {i, 1, x}];(* 能力重視の選抜 *)
  a2 = Sort[all, #1[[2]] > #2[[2]] &];(* 語学力によるソート *)
  new2 = Table[a2[[i]][[1]], {i, 1, x}];(* 語学力重視の選抜 *)
  Print[{"能力重視した場合の生産性=", Mean[new1],
 "語学重視した場合の生産性=", Mean[new2],  "p-value=", TTest[{new1, new2}]}];
  ListPlot[a1, PlotRange -> {{30, 70}, {30, 70}}]  ]
```

問題．倍率が上がると，生産性の低下がより大きくなるかどうか確認せよ．

問題．2 段階選抜のモデルを考えよ．語学→生産能力の場合と生産能力→語学の場合とでは結果が異なるかどうかを確認せよ（jobint2.nb 参照）

問題．生産力と語学力の合計得点で選抜した場合の，生産能力の平均値を比較せよ

以上の数値計算による分析結果を数理モデルで表現しておこう．2 種類の能力が 2 次元正規分布に従うという仮定から，2 つの確率変数 X_1, X_2 の同時確率密度関数は

$$f(x_1,x_2)=\frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}}\exp\left\{-\frac{1}{2(1-\rho^2)}\left(\frac{(x_1-\mu_1)^2}{\sigma_1^2}+\frac{(x_2-\mu_2)^2}{\sigma_2^2}-\frac{2\rho(x_1-\mu_1)(x_2-\mu_2)}{\sigma_1\sigma_2}\right)\right\}$$

である．2 つの確率変数が 2 次元正規分布に従うことを記号で

$$(X_1,X_2)\sim N\left((\mu_1,\mu_2),\begin{pmatrix}\sigma_1^2 & \sigma_{12}\\ \sigma_{12} & \sigma_2^2\end{pmatrix}\right)$$

と書く．生産能力 X_1，語学力 X_2 の上位 10%タイルをそれぞれ $x_1(0.1)$，$x_2(0.1)$ で表す．x_1 の周辺確率密度を

$$h(x_1)=\int_{-\infty}^{\infty} f(x_1,x_2)\,dx_2$$

とおけば，生産能力基準で上位 10%を採用した場合の生産性平均値は

$$E[X_1 \mid X_1 > x_1(0.1)]=\int_{x_1(0.1)}^{\infty} x_1\cdot\frac{h(x_1)}{0.1}\,dx_1=10\int_{x_1(0.1)}^{\infty}\left(x_1\cdot\int_{-\infty}^{\infty} f(x_1,x_2)\,dx_2\right)dx_1$$

である．一方，語学力基準で上位 10%を採用した場合の生産性平均値は

$$E[X_1 \mid X_2 > x_2(0.1)]=\int_{-\infty}^{\infty} x_1\cdot\varphi(x_1)\;dx_1$$

$$=\int_{-\infty}^{\infty}\left(x_1\cdot\int_{x_2(0.1)}^{\infty} f(x_1,x_2)\,dx_2\right)dx_1$$

である．シミュレーションに基づく予想が正しいとするならば，

$$(X_1,X_2)\sim N\left((\mu_1,\mu_2),\begin{pmatrix}\sigma_1^2 & \sigma_{12}\\ \sigma_{12} & \sigma_2^2\end{pmatrix}\right),0<\rho<1\Rightarrow\; E[X_1 \mid X_1 > x_1(0.1)]\geq E[X_1 \mid X_2 > x_2(0.1)]$$

が成立する[1].

問題．上記の不等式を証明せよ．

[1] シミュレーションによる分析は簡単で有用だが，一般性はない．一方で上記のような数学的アプローチは，難しいがひとたび結果が証明できれば，一般的であり，（ほぼ確実に）未来永劫正しい．数理社会学の最終目標は数学的に証明可能な一般命題を得ることである． シミュレーションと違い，解析的アプローチは必ず結果が出るとは限らないため（証明できるとはかぎらない），それを覚悟のうえで取り組まなくてはならない．

証明の容易な離散確率変数の場合について解説する（連続確率変数の場合については毛塚君による証明を参照）.

命題．二つの能力がベクトル x_1 と x_2 で表されている．二つのベクトルは同じ数の要素を含んでいる.

$$x_1 = (x_{11}, x_{12}, ..., x_{1n})$$
$$x_2 = (x_{21}, x_{22}, ..., x_{2n})$$

各上位 10 パーセンタイルを $x_1(0.1), x_2(0.1)$ とおく．このとき
「$x_{1i} \geq x_1(01)$ を満たす x_{1i} の平均 $E(X_1 \mid X_1 \geq x_1(0.1))$」と，
「$x_{2i} \geq x_2(01)$ を満たす順序対 (x_{1i}, x_{2i}) の第一次元の要素 x_{1i} の平均 $E(X_1 \mid X_2 \geq x_2(0.1))$」を比較すると

$$E(X_1 \mid X_1 \geq x_1(0.1)) \geq E(X_1 \mid X_2 \geq x_2(0.1))$$

が成立する.

証明．各ベクトルを

$$x_1 = (x_{11}, x_{12}, ..., x_{1n})$$
$$x_2 = (x_{21}, x_{22}, ..., x_{2n})$$

とおく． $x_{1i} \geq x_1(01)$ を満たす x_{1i} の集合を A とおく.

$$A = \{x_{1i} \mid x_{1i} \geq x_1(0.1)\}$$

また， $x_{2i} \geq x_2(01)$ を満たす順序対 (x_{1i}, x_{2i}) の第一次元の要素 x_{1i} の集合を B とおく.

$$B = \{x_{1i} \mid x_{2i} \geq x_2(0.1)\text{を満たす}(x_{1i}, x_{2i})\text{の}x_{1i}\}$$

集合 A, B が一致する場合

集合 A, B が一致するとき，平均値も一致するので

$$E(X_1 \mid X_1 \geq x_1(0.1)) = E(X_1 \mid X_2 \geq x_2(0.1))$$

である.

集合 A, B が一致しない場合

一般性を損なうことなく，$|A| = |B| = m$ と仮定し，それぞれ昇順に並べたうえで次のように番号を付け替える．ここで a_i, b_i は全て x_1（生産能力ベクトル）の要素である事に注意する.

$$A=\{a_1,a_2,...,a_m=x_1(0.1)\},\quad a_1>a_2>\cdots>a_m=x_1(0.1)$$
$$B=\{b_1,b_2,...,b_m\},\quad b_1>b_2>\cdots>b_m$$

昇順に並べているため，集合 A,B が一致しないということは，ある番号 $j\geq 1$ が存在して，

$$a_1=b_1,a_2=b_2,...,a_{j-1}=b_{j-1}$$

j 番より前の要素は互いに等しく， j 番については

$$a_j>b_j$$

が成立する（ただし逆の $a_j<b_j$ は a_i が x_1 の要素を昇順に並べているという仮定に反する）．

a_i が x_1 の要素を昇順に並べているという仮定から， $j+1$ 番目について，

$$a_j>a_{j+1}\geq b_j>b_{j+1}$$

だから $a_{j+1}>b_{j+1}$ である．それ以降についても同じ比較を順次適応すれば

$$a_{j+1}>b_{j+1},a_{j+2}>a_{j+2},...a_m>b_m$$

が成立する．よって集合 B の j 番目以降の要素は A の j 番目以降の要素よりも小さいので，

$$E(X_1\mid X_1\geq x_1(0.1))>E(X_1\mid X_2\geq x_2(0.1))$$

である．

集合 A,B は一致するか一致しないかのどちらかだから，まとめると

$$E(X_1\mid X_1\geq x_1(0.1))\geq E(X_1\mid X_2\geq x_2(0.1))$$

が成立する．

5．実証分析の功罪　　olssim1.nb

近年では R をはじめとする無料統計ソフトの普及により，誰でも簡単にデータ分析ができるようになった．しかし社会(科)学者の多くは，統計モデルを適用できる分析の範囲を過大に解釈しているようである．言い換えれば，必要な仮定が満たされていない場合でも，それを無視していることがしばしばある．当然ながらそのような分析および分析で推定された推定値は信用できない．以下では OLS（を基礎とする一般化線型モデル）が「真の関数」の近似として，どの程度有功なのかを簡単なシミュレーションで確認する．

手順

- 線型関数ではない「真の関数」を仮定して，説明変数に適当な固定数値を代入する．その結果に攪乱項（誤差項）を追加して，被説明変数と定義する．
- 真の関数に代入した説明変数に対して，（攪乱項込みの）被説明変数を回帰する．
- OLS 推定したパラメータを仮説検定する．

5.1. 関数の定義

```
(*  攪乱項（誤差項）を含む真の関数からの出力  *)
er[s_] := RandomReal[NormalDistribution[0, s]];(* 攪乱項の定義．平均は 0 *)
truef1[x1_, x2_, x3_] := 20 + 5 x1 + 10 x2 - 15 x3 + er[1];
truef2[x1_, x2_, x3_] := 20 + 10 x1*x2*x3 + er[1];
truef3[x1_, x2_, x3_] := 20 + 5 x1^2 + 10 x2^3 + Log[x3] + er[1];
truef4[x1_, x2_, x3_] := 20 + Sqrt[5 x1]*Sqrt[x2]*Log[x3] + er[1];
truef5[x1_, x2_] := Sqrt[5 x1]*x2^5 + er[1];
freal6[x1_, x2_, x3_] := 20 + 10 x1*x2*x3 + er[1];
```

5.2. 分析例

```
Module[{population, data, lm},
 population =  Table[{x1, x2, x3, truef2[x1, x2, x3]},
 {x1, 1, 10}, {x2, -1, 5}, {x3, 1, 10}];
 population = Flatten[population, 2];(*説明変数により異なる、注意*)
 population = RandomSample[population];
 data = population[[1 ;; 100]](* 無作為抽出 *);
 lm = LinearModelFit[data, {x1, x2, x3}, {x1, x2, x3}];
```

```
(* ここで線型回帰 *)
 Normal[lm];
 TableForm[
  lm[{"ParameterTable", "ANOVATable", "AdjustedRSquared", "AIC"}]
  ] ]
```

以上の計算結果から，推定したパラメータの符号（正負）が，真のパラメータの符号と異なる場合さえあることが分かる．真の関数が線形でないときも，決定係数が高いことは，十分にありえることを確認しておこう．

5.3. 科学的説明における実証分析の解釈

Q　実証分析の結果変数 X のパラメータが正であり，統計的に有意であったとしよう．その結果に基づいて，「『X が増加すると Y が増加する』という関係が，（高い確率で）経験的に成立している」と主張することは許されるだろうか？

参考資料　TeX　2015 数理行動科学演習演習・行動科学演習

６．エラーへの対処

よく生じるエラーメッセージを以下にあげておく．計算結果がうまくでないときは，次の項目を確認してみよう

１．セルの右側に「+」マークが表れ，計算結果が出てこない．

「+」マークをクリックすると，エラー内容が表示される..

考えられる原因

[　]の閉じ忘れ.

［　］の対応確認方法

　]　の右側にカーソルがある状態で，「Ctrl」+「Shift」+「B」を押す．すると対応している　[　までが反転表示される.

２．赤い　^（ハット）が表示され，正しい結果がでない．

考えられる原因

;　（セミコロン）や　，（カンマ）が正しい位置にない.

３．Set::write:XXXX のタグ Times は Protected です．>>

考えられる原因

コマンドを複数行書いたときにセミコロン（;）を書き忘れている．一つの処理が終わった場合は必ずセミコロンで終わること.

例えば次のような Module を実行するとこのエラーが出る.（；）が必要な位置に抜けているため，リストの乗算と解釈され，「そんなことできないよ」というエラーが出る.

```
Module[{x, y},
 x = Table[i, {5}]
 y = Table[1, {5}];
 x + y]
```

4. Thread::tdlen: {XXX}+{XXXX}**で互いに長さが等しくないオブジェクト同士は結合できません．** >>

考えられる原因

リストの演算で，結合させるリスト同士の長さがあっていない．

例えば次のような Module を実行するとこのエラーが出る．要素数が 5 のリストと 6 のリストを足そうとするとエラーが出る．

```
Module[{x, y},
 x = Table[i, {5}];
 y = Table[i, {6}];
 x + y]
```

Thread::tdlen: {i,i,i,i,i}+{i,i,i,i,i,i}**で互いに長さが等しくないオブジェクト同士は結合できません．** >>

5. Set::setraw: **未加工オブジェクト** XXX **に割当てができません．** >>

考えられる原因

関数の引き数を Module 内で再代入している．このメッセージは，割当ての左辺が数または文字列の場合に出力される（関数の引数は数扱い）．

例えば次のような Module を実行するとこのエラーが出る．

```
f[x_] := Module[{s},
  s = 5;
  x = x + s];
f[10]
```

Set::setraw: **未加工オブジェクト** 10 **に割当てができません．** >>

引数の値を更新したい場合は次のように書けばよい．

```
f[x_]:=Module[{s,c=x},
  s=5;  c=c+s]
```

これなら問題なく

```
f[10]
```
という入力に対して
```
15
```
という結果が返ってくる

6．Part::partw: {XXXX}の部分 XX は存在しません． >>

考えられる原因

リストの要素を抽出する際，リストの長さを超える位置を指定している．自分ではリストだと思っているオブジェクトが実際にはリストではない場合に，要素抽出すると，このエラーがしばしば起こる．

例えば次のような命令を実行するとこのエラーが出る．

```
In[1]= x={1,2,3,4,5};
x[[6]]
Part::partw: {1,2,3,4,5}の部分 6 は存在しません． >>
Out[1]={1, 2, 3, 4, 5}[[6]]
```

7．〇〇は〇〇に似ています>>

考えられる原因

変数名が，定義済みの内部関数と同じ名前になっている．大文字から始めるべき内蔵関数が小文字から始まっている．

8．〇〇は１個の引数で呼ばれました. 2 つの引数が想定されています． >>

考えられる原因

引数の数が，定義と違っている．本来 2 つの引数が必要なのに，1 個しか指定しないと，このエラーがでる．

9．Power::infy: 無限式 1/0 が見付かりました． >>

考えられる原因

不定式（分母が 0 になっている式）が含まれている．手続き上，0 が分母に入る場合は，If 文を使って不定形を回避する．

Q　これまでの実習で目にしたエラー文をひとつあげよ．そのエラーがどんな原因で生じたかを考察して，デバッグせよ．

一般に，プログラミング作業の 7〜9 割はデバッグである． はじめのうちは，なぜエラーが出るのか理解できず，その修正に膨大な時間を要する．しかし慣れるに従って，対処に要する時間が短くなり，エラーの発生頻度も減少するので，くじけずにトライしてほしい．デバッグの手順は

1. どこでエラーが出たのかをまず特定する（`Print` やデバッガ機能を活用する）
2. エラー部分だけを切り出して修正する
3. 修正した部分を元に戻す

7．アルゴリズムとはなにか？

アルゴリズムという言葉は，9 世紀にバグダードで活躍したアラビアの科学者アル・フワーリズミーのラテン語名の変化形である Algoritmi に由来する．代数を意味するアルジェブラ Algebra は，彼の著書『アル=ジャブルとアル=ムカバラの計算』のアル=ジャブル（移項）に由来する．（『数学入門辞典』）

Q　100 以下の素数を列挙するアルゴリズムを考え，それをプログラムで表現せよ．

あらゆるプログラムの基本

- 与えられた問題を解くための手順を「普通の言葉」で考える
- 「普通の言葉」で考えた手順をプログラム言語に翻訳する
  - 前掲，特によく使う関数表参照
  - 関数を知らない場合は，`For, Do, If, Table` を利用して作る．
- 具体的な数値例をもとにプロトタイプを作る
- プロトタイプをもとに一般化する

例として 30 より小さい素数がいくつかあるか，という問題を考える．

定義（素数と合成数）．1 よりも大きく，かつ，『1』と『それ自身』でしか割り切れない整数を素数という．1 より大きい整数で素数ではないものを合成数という．

例．2，3，5，7 は素数である．しかし 4 や 6 は素数ではなく合成数である．なぜなら 4 は 2 で割り切れるし，6 は 2 でも 3 でも割り切れる．

なお 1 は素数ではない．　1 は『1』と『それ自身』でしか割り切れない，という性質を満たしているが，1 を素数に含むと，素因数分解の一意性が成り立たなくなるためである．

7.1.　エラトステネスのふるい

まず 30 以下の数をならべる．1 は定義からいって素数にならないので，のぞいておく．

2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30

一番最初の数の倍数を列から消し，最初の数だけを素数としてマークする．この場合，一番左の 2 をマークして 2 の倍数を列から消す（2 の倍数をふるいにかけて，2 の倍数ではない数だけを残す）.

2, 3, ~~4,~~ 5,~~ 6,~~ 7, ~~8,~~ 9, ~~10~~, 11, ~~12~~, 13, ~~14~~, 15, ~~16~~, 17, ~~18~~, 19, ~~20~~, 21, ~~22~~, 23,~~ 24~~, 25,~~ 26~~, 27, ~~28~~, 29,~~ 30~~

↓　2 の倍数に線を引き，2 の倍数を消した.

2, 3, 5, 7, 9, 11, 13, 15, 17, 19,　21, 23, 25, 27,　29,

次に，2 の次にある 3 をマークして 3 の倍数を消す.

2, 3, 5, 7, ~~9,~~ 11, 13, ~~15,~~ 17, 19, ~~21,~~ 23, 25, ~~27~~, 29

↓　3 の倍数に線を引き，3 の倍数を消した.

2, 3, 5, 7, 11, 13, 17, 19, 23, 25, 29

次に，残った数字の列から 3 の次にある 5 をマークして 5 の倍数を消す.

2, 3, 5, 7, 11, 13, 17, 19, 23, ~~25,~~ 29

↓　5 の倍数に線を引き，消した.

2, 3, 5, 7, 11, 13, 17, 19, 23, 29

次に，残った数字の列から 5 の次にある 7 をマークして 7 の倍数を消す.

2, 3, 5, 7, 11, 13, 17, 19, 23, 29

この中に 7 以外には 7 の倍数は残っていない．その場合は次の数字の倍数を消す作業にとぶ．7 の次は 11 なので 11 の倍数をさがす.

2, 3, 5, 7, 11, 13, 17, 19, 23, 29

11 の倍数も残っていない. 次の 13 の倍数も残っていない. 17 も同様. 残りの数については，その倍数が残っていない．結局

2, 3, 5, 7, 11, 13, 17, 19, 23, 29

これで 30 より小さい素数全てを取り出したことになる.

これで n 以下の素数の求め方が分かる．手順を書けばこうだ．

1. 2 以上 n 以下の整数を全て並べる．
2. 2 を残して 2 の倍数を消す．
3. 3 を残して 3 の倍数を消す．
4. 以上を繰り返すと n 以下の全ての素数が残る．

直感的には，100 以下の素数を数え上げたければ，2 からはじめて 99 の倍数まで，順番に消していけばいい，と考えたくなるが，実際には 100 の平方根である 10 まで考えれば十分である．なぜなら合成数はその平方根以下の数に分解されるからである．

定理（$\sqrt{n}$ 以下の素因数）1 より大きい n について，$\sqrt{n}$ 以下の素因数（素因数分解した時の素数部分）が存在しなければ，n は素数である（Hungerford 1997）．
証明．練習として各自で考えてみよう

7.2. コマンドへの変換

- 「普通の言葉」で考えた手順をプログラム言語に翻訳する

次の手順に分割する．

1. はじめに 2 から 30 までのリストを作る→`Range` **あるいは** `Table`
2. リストの最初の数をとりだして，その数の倍数のリストをつくる．→`Table`
3. もとのリストから倍数を取り除く→ `Complement`

- 具体的な数値例をもとにプロトタイプを作る
  sieve.nb を参照

- プロトタイプをもとに一般化する

実例として，http://demonstrations.wolfram.com/SieveOfEratosthenes/

7.3. もう一つの解答例

内蔵関数 `Prime` をつかう．

```
Table[Prime[i],{i,1,10}]
```

この命令は，`10` 個の素数を昇順でならべる．

8．ランチェスターの法則　battle.nb

次のような状況をプログラムで表現してみよう．

グループ 1 とグループ 2 の戦士が招集される．初期条件として各兵士の武力は正規分布にしたがう．兵士達は敵国の兵士と，騎士道精神に則り一対一で戦う．

ソース例

```
Module[{id1, id2, n1 = 20, n2 = 20, power1, power2, winner1, winner2},
 id1 = Range[n1];(*  グループ１の兵士の ID を定義  *)
 id2 = Range[n2];(*  グループ２の兵士の ID を定義*)
  (* 各グループの戦闘力を定義*)
 power1 = Table[RandomReal[NormalDistribution[50, 1]], {n1}];
 power2 = Table[RandomReal[NormalDistribution[51, 1]], {n2}];
 winner1 = {}; winner2 = {};(* 勝者の番号を格納するベクトル *)
  (* 戦闘力を比較して勝者を残す*)
 Table[  If[
   power1[[id1[[i]]]] > power2[[id2[[i]]]],
   winner1 = Append[winner1, id1[[i]]],
   winner2 = Append[winner2, id2[[i]]]
   ](* If 終了 *),
  {i, 1, Min[n1, n2]}](* Table *);

 Print[{グループ１戦闘後, winner1}];
 Print[{グループ２戦闘後, winner2}];
 Print[{"戦闘力の組み合わせ",
   MatrixForm[
    Table[{i, N[power1[[i]], 3], N[power2[[i]], 3]}, {i, 1,
      Min[n1, n2]}]]}]
 ]
```

出力例

{グループ１戦闘後,{9,17,19}}

{グループ２戦闘後,{1,2,3,4,5,6,7,8,10,11,12,13,14,15,16,18,20}}

{戦闘力の組み合わせ,

1　　48.71　　50.1785

2　　50.0097　　52.8188

＜中略＞

19　　50.9768　　49.8753

20　　49.6952　　51.9615

8.1. 戦闘の反復

```
Module[{id1,id2,n1=10,n2=10,power1,power2,winner1,winner2,reserve1,reserve2},
power1=Table[RandomReal[NormalDistribution[10,1]],{n1}];(*初期兵力*)
power2=Table[RandomReal[NormalDistribution[11,1]],{n2}];(*初期兵力*)
id1=Range[n1];(*初期兵 ID*)id2=Range[n2];(*初期兵 ID*)

While[(Length[id1]>0)&&(Length[id2]>0),(*戦闘終了条件　生存者がいる限り反復　*)
winner1={};winner2={};(*戦闘勝者リストの初期化*)

id1=RandomSample[id1];Print[{"G1:出撃する兵士の ID",id1}];(* 戦闘の度に相手が変わる *)
id2=RandomSample[id2];Print[{"G2:出撃する兵士の ID",id2}];

Table[ If[
        power1[[id1[[i]]]]>power2[[id2[[i]]]],
        winner1=Append[winner1,id1[[i]]],(* 勝者リストに Append で加える*)
        winner2=Append[winner2,id2[[i]]] (* 勝者リストに Append 加える*)
        ](* ここまで If 文 *),{i,1,Min[Length[id1],Length[id2]]}]
        (* Table ここまで*);

Print[{"G1:戦闘で勝った兵士",winner1}];
Print[{"G2:戦闘で勝った兵士",winner2}];
```

```
reserve1=If[
Length[id1]>Length[id2],
Table[id1[[i]],{i,Length[id2]+1,Length[id1]}],
{}](* ここまで If 文 *);
reserve2=If[
Length[id2]>Length[id1],
Table[id2[[i]],{i,Length[id1]+1,Length[id2]}],
{}];(* 戦闘に参加しなかった予備役を招集する *)

id1=Append[winner1,reserve1]//Flatten;
id2=Append[winner2,reserve2]//Flatten;
Print[{"G1:残存兵",id1}];
Print[{"G2:残存兵",id2}]](* while ここまで*)
](* Module 閉じる *)
```

反復することで，戦闘力の高い兵士が生き残る様子が観察できる．

関数化の例．

```
Battle[n1_,(*集団１の人数*)n2_,(*集団２の人数*) m1_,(*平均１*)m2_,(*平均２*),s1_(*標準
偏差１*),s2_(*標準偏差２*)]:=
Module[{id1,id2,power1,power2,winner1,winner2,reserve1,reserve2},
power1=Table[RandomReal[NormalDistribution[m1,s1]],{n1}];(*初期兵力*)
power2=Table[RandomReal[NormalDistribution[m2,s2]],{n2}];(*初期兵力*)
＜途中省略＞
id1=Append[winner1,reserve1]//Flatten;
id2=Append[winner2,reserve2]//Flatten;
Print[{"G1:残存兵",id1}];
Print[{"G2:残存兵",id2}]](* while ここまで*)
](* Module 閉じる *)
```

この関数は引数は n1,n2,m1,m2,s1,s2,である．引数が多いと，入力の際に順番を混同しやすいので，関数を定義するときに引数にコメントを使うとよい．

Q　戦闘を繰り返す度に，疲弊して戦力が低下する状況を表現してみよう

Q　1 対 1 から，1 対多の近代戦を表現できるか？　考えてみよう

　　→　長さの異なるベクトルをどうやって対応させるか？

Q　自分が表現したい戦いを自由に考え，コード化してみよう

8.2.　数理モデルによる一般化

このシミュレーションは，二組の確率変数の実現値をランダムマッチングして差をとる，という操作を実行している．X という確率変数の実現値のベクトルが一方にあり，Y という確率変数の実現値のベクトルが一方にある．この二つのベクトルの引き算が，求めた結果である．この結果は実現値と組み合わせによって毎回違うが，1 回の試行は「確率変数 X と Y を合成して作った確率変数 $Z = X - Y$ の分布がどのような分布に従うのか」を計算しているに過ぎない（厳密に言えば，Z の分布を作ってから，さらに非負の値だけ取り出した者が，生き残った兵の分布である）．

この分布は以下のように理論的に特定できる（以下の計算は正規分布が再生性を持つ，という非自明な命題の初等的な証明にも対応している．またこの合成積という手法は多くの確率変数の合成に応用できるので覚えておくと便利である）．

補題.

$$\int_{-\infty}^{\infty} \exp\left\{-\frac{(x+b)^2}{a}\right\} dx = \sqrt{a\pi}$$

命題(正規分布の差による合成)．$X \sim N(\mu_x, \sigma_x^2), Y \sim N(\mu_y, \sigma_y^2)$ とおく．　$W = X - Y$ の分布は

$$W \sim N(\mu_x - \mu_y,\ \sigma_x^2 - 2\rho\sigma_x\sigma_y + \sigma_y^2)$$

である．特に，X, Y が独立なとき（つまり相関が 0 であるとき），

$$W \sim N(\mu_x - \mu_y, \sigma_x^2 + \sigma_y^2)$$

である．

証明．一般にどんな確率変数でも，独立であるかないかにかかわらず

$$E(X \pm Y) = E(X) \pm E(Y), V(X \pm Y) = V(X) + V(Y)$$

が成立する．しかし，確率密度関数が同じであるかどうかは自明ではない．以下合成積を使って直接計算で示す.

確率変数 X, Y の合成を次のように考える.

$$\begin{cases} w = x - y \\ z = y \end{cases} \qquad \begin{cases} x = w + z \\ y = z \end{cases}$$

このとき z はどのように定義してもよいが, なるべくヤコビアンが単純になるような定義を使うとよい．確率変数 $W = X - Y$ の確率密度関数 $g(w)$ は，W と Z の同時密度

$$\varphi(w,z) = f(x,y)\begin{vmatrix} \frac{\partial x}{\partial w} & \frac{\partial x}{\partial z} \\ \frac{\partial y}{\partial w} & \frac{\partial y}{\partial z} \end{vmatrix} = f(x,y)\begin{vmatrix} 1 & 1 \\ 0 & 1 \end{vmatrix} = f(x,y)\cdot 1 = f(w+z,z)$$

より

$$g(w) = \int_{-\infty}^{\infty} \varphi(w,z)\, dz = \int_{-\infty}^{\infty} f(w+z,z)\, dz$$

である[2]．明示的に書くと (X,Y) の同時確率密度関数の定義から

$$g(w) = \int_{-\infty}^{\infty} \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left\{-\frac{1}{2(1-\rho^2)}\left(\frac{(w+z-\mu_x)^2}{\sigma_x^2} + \frac{(z-\mu_y)^2}{\sigma_y^2} - \frac{2\rho(w+z-\mu_x)(z-\mu_y)}{\sigma_x\sigma_y}\right)\right\} dz$$

である．指数部を

$$\frac{(w+z-\mu_x)^2}{\sigma_x^2} + \frac{(z-\mu_y)^2}{\sigma_y^2} - \frac{2\rho(w+z-\mu_x)(z-\mu_y)}{\sigma_x\sigma_y} = az^2 + bz + c$$

とおいて z に関して平方完成する．a, b, c は明示的には，それぞれ

$$a = \frac{\sigma_x^2 - 2\rho\sigma_x\sigma_y + \sigma_y^2}{\sigma_x^2\sigma_y^2}, b = \frac{2\{(\mu_x + \mu_y - w)\rho\sigma_x\sigma_y + (w-\mu_x)\sigma_y^2 - \mu_y\sigma_x^2\}}{\sigma_x^2\sigma_y^2},$$

$$c = \frac{\mu_y^2\sigma_x^2 + 2(w-\mu_x)\mu_y\rho\sigma_x\sigma_y + (w-\mu_x)^2\sigma_y^2}{\sigma_x^2\sigma_y^2}$$

---

[2] 一変数の置換積分法の場合 $x = g(t)$ とおいて $\int f(x)dx = \int f(g(t))g'(t)dt$ となる．この $g'(t)$ がヤコビアンだと（アナロジーで）考えると直感的に理解しやすい.

である．定数部分を $\left(2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}\right)^{-1} = K$ とおけば，

$$
\begin{aligned}
g(w) &= K\int_{-\infty}^{\infty}\exp\left\{-\frac{1}{2(1-\rho^2)}\left(a\left(z+\frac{b}{2a}\right)^2-\frac{b^2}{4a}+c\right)\right\}dz \\
&= K\exp\{D\}\int_{-\infty}^{\infty}\exp\left\{-\frac{1}{2(1-\rho^2)a^{-1}}\left(z+\frac{b}{2a}\right)^2\right\}dz \\
&= K\exp\{D\}\int_{-\infty}^{\infty}\exp\left\{-\frac{(z+A)^2}{2C}\right\}dz = K\exp\{D\}\sqrt{C}\sqrt{2\pi}
\end{aligned}
$$

ただし，

$$
A=\frac{b}{2a}\ ,C=(1-\rho^2)a^{-1},\ D=-\frac{1}{2(1-\rho^2)}\left(-\frac{b^2}{4a}+c\right)
$$

である． C,D,K を代入して

$$
g(w)=K\exp\{D\}\sqrt{C}\sqrt{2\pi}=\frac{1}{\sqrt{2\pi}(\sigma_x^2-2\rho\sigma_x\sigma_y+\sigma_y^2)}\exp\left\{-\frac{(w-(\mu_x-\mu_y))^2}{2(\sigma_x^2-2\rho\sigma_x\sigma_y+\sigma_y^2)}\right\}
$$

を得る． $g(w)$ は確率変数 $W=X-Y$ の確率密度関数である．よって

$$
W\sim N(\mu_x-\mu_y,\ \sigma_x^2-2\rho\sigma_x\sigma_y+\sigma_y^2)
$$

が示せた． $W=X+Y$ と仮定した場合の分布も同様の計算によって特定できる．

9. 調整ゲームのレプリケータダイナミクス

車の通行に関して明示的なルールが存在しない社会を想定する（例えば車が発明された直後の世界）．ある道で，すれ違おうとする 2 人のドライバーA と B が「右側を走る」か「左側を走る」かを考えている．2 人とも「右側通行」あるいは「左側通行」の場合，衝突は起こらず，問題はない．しかし一方が右側通行をして他方が左側通行をすると，衝突事故が起こる．この状況を 「利得行列」 という形式で簡単に表現することができる．

		プレイヤーB	
		右側通行	左側通行
A	右側通行	（2 ， 2）	（0 ， 0）
	左側通行	（0 ， 0）	（1 ， 1）

利得行列という表現によって （右，右） か （左，左） が実現すれば，それがナッシュ均衡 であることが分かる．しかしこの二つの均衡のうち，どちらが選ばれるかは分からない．そこで進化ゲーム理論の考え方を導入する．

ある時点で，社会の中にさまざまな戦略を持った個体が存在すると仮定する．
各個体は他の個体とペアになり，ゲームを行い，ゲームの結果に応じて利得を得る．
その利得に基づいてそれぞれの戦略を持った個体の割合が変化する．

流れ：

- 初期値として各個体の右戦略の選択確率を定義
- 個体間でランダムマッチングしてペアが形成される （1）
- 確率に基づき，戦略を決定する
- ペアになった個体同士でゲームを行う
- 実現した戦略の組み合わせで各個体の利得が決まる
- 利得に応じて次回の戦略選択確率を変化させる（利得が高い戦略ほど選択されやすい）
- （1）に戻る

以上にプログラムの重要箇所を解説する．

マッチングのコード

```
x = RandomSample[Range[n]];
p = Range[n];
Table[p[[x[[i]]]] = x[[n + 1 - i]], {i, 1, n}];
partner = p;(* 繰り返しの度にマッチング Print[partner];*)
```

1. x **にランダムにシャッフルした番号を並べる**
2. p **を昇順に並べる.**
3. p **の** x[[i]]**番目に,** x[[n + 1 - i]](x **の後ろから** i **番目の人)を代入する.**
4. p **マッチング済みのリスト** p **を変数** partner **に代入する**

戦略決定のコード

```
strategy = Table[If[prob[[i]] > RandomReal[], 1, 0], {i, 1, n}];
```

1. **初期確率を定義したリスト** prob **の値が, 一様乱数よりも大きい場合に** 1**, そうでない場合に** 0 **を入力する.**
2. **より単純には** prob[[i]]>0.5 **でもよい**

利得決定

```
payoff = Table[
  Which[
   (strategy[[i]] == 1) && (strategy[[partner[[i]]]] == 1), a,
   (strategy[[i]] == 1) && (strategy[[partner[[i]]]] == 0), 0,
   (strategy[[i]] == 0) && (strategy[[partner[[i]]]] == 1), 0,
   (strategy[[i]] == 0) && (strategy[[partner[[i]]]] == 0), b],
{i,1, n}];
```

1. Table **を使って,** Which **を全員に適応する**
2. Which **は戦略の組み合わせに応じて, 利得リスト** payoff **に, 各個人が獲得する利得を代入する.**
3. &&**は「かつ」を意味する. 例えば** 3 **行目の命令は「自分の戦略が** 1**」かつ「相手の戦略が** 1**」ならば** a **を代入する.**

更新ルールの選択

レプリケータダイナミクス系の話では, 戦略の更新法にいくつかの種類がある. 基本的には高い

利得を得る戦略を選択しやすくなる, というルールを採用する.

```
   Table[   Which[
     ((strategy[[i]] == 1) && (payoff[[i]] > groupmean)),
     prob[[i]] = prob[[i]] + (a/(a + b) (1 - prob[[i]])),
     ((strategy[[i]] == 0) && (payoff[[i]] > groupmean)),
     prob[[i]] = prob[[i]] - (b/(a + b) (prob[[i]])),
     payoff[[i]] == 0, prob[[i]] = prob[[i]]],
    {i, 1, n}],

   Table[  Which[
     ((strategy[[i]] == 1) && (payoff[[i]] > groupmean)),
     prob[[i]] =
      prob[[i]] + 0.1 (payoff[[i]]/(payoff[[i]] + groupmean)),
     ((strategy[[i]] == 0) && (payoff[[i]] > groupmean)),
     prob[[i]] =
      prob[[i]] - 0.1 (payoff[[i]]/(payoff[[i]] + groupmean))],
    {i, 1, n}]];
```

Manipulate について

計算結果の可視化のために関数 Manipulate を使う.

```
Manipulate[
 Dynamic[
  Refresh[
   ListPlot[cogame[n, a, b, time, rule], PlotRange -> {0, 1}],
   UpdateInterval -> If[moving, 0, Infinity]](*reflesh*)
  ](*dynamic*),
 {{moving, False, "run simulation"}, {True, False}},
 {rule, {1, 2}},
 {{a, 2, "右側通行の利得"}, Range[0, 10]},
 {{b, 2, "左側通行の利得"}, Range[0, 10]},
 {{time, 50, "繰り返し数"}, 10, 100, 1, ImageSize -> Tiny,
```

```
 Appearance -> "Labeled"},
{{n, 100, "エージェント数"}, 50, 1000, 5, ImageSize -> Tiny,
 Appearance -> "Labeled"}, ControlPlacement -> Left
]
```

問題．一般の 2*2　対称ゲームの利得行列を分析できるようにプログラムを一般化せよ（チキンゲームや囚人のジレンマを一つのプログラムで分析する）

9.1. Module の使い方

関数の定義は　例えば二次関数 $y = ax^2 + bc + c$ ならば

```
f1[a_,b_,c_,x_]:=a x^2 + b x + c
```

のように書く．一般的には

関数の名前[変数 1_,変数 2_,変数 3_]:=手続き

となる．独立変数にはアンダーバーを忘れずに入れる．また定義した関数は，Shift+Enter キーで読み込まないと使えない．定義を修正した場合も，逐一再読み込みしなくてはならないので注意する．関数化することの利点は，あとで数値を変更して再分析することが容易になる点にある．

Q.　二次関数のグラフを Plot してみよ

A.　`Plot[ f1[1,2,2,x], {x, -5,5}]`

Module は一連の手続きをまとめて，その手続きを関数化するために使う方法である．Module を組み合わせることで，複雑なプログラムも容易に書ける．

例えば次のような単純な関数ならばダイレクトに定義できる

```
S2[s_, r_] := Tuples[Table[Range[r], {i, s}]] // Reverse;
```

しかし，いくつかの局所変数を使い，複数の手順を経て計算する場合は Module でまとめる必要がある．

例：実行には `Needs["BarCharts`"];`　が必要

```
        関数の引数              局所変数
            ↓                    ↓
fkimaged[s_, r_] := Module[{m, x, im},
  m = S2[s, r];
  x = Table[Plus @@ m[[i]] - (s - 1), {i, 1, Length[m]}];
  im = BinCounts[x, {1, s (r - 1) + 2(*イメージ階層数+1*)}];
  BarChart[im]   ]
```

- 関数の引数とはモデルを分析する際に様々な値を入力して比較するための変数である（説明変数と考えればよい）．一方，Module 内の局所変数は手続きをまとめるための一時的な名前で，計算の度に値が異なるので，実行後には残らない．
- Module 内の局所変数名と関数の引数の名前が重複するとエラーが生じるので注意する．
- Module を使った関数の定義の手順は次の通りである．
  - Module[{a=1,b=3,x},手続き 1;手続き 2] のように初期値を与えながら Module だけを先に完成させる．
  - 次に f[a_,b_]:=Module[{x},手続き 1;手続き 2] のように分析のために動かしたいパラメータを関数の引数として外に出す．以上．
- 一端出来上がった関数を修正する場合は，丸ごとコピーして下のセルに貼り付け，名前だけを変更する（例えば f1,f2,f3 のように順次数字を変える）．こうすることで，より簡単に追加した仮定の効果を知ることができるし，確実に作業が前進する．
- セルの「折り畳み」を利用すればプログラムを簡単に整理することができる
- 部分的な結果の抽出には Print[]を使う．上の関数で x や m を Print してみると途中の経過が分かる．Mathematica でデバッグするときは基本的には Print を使う．コピーしたあとで，途中を切ってもよい．
- Module を使う場合には命令の区切りに ; を入れるのを忘れないよう気をつける．
- コマンドの一部分を試してみたい場合は，下の空白セルを利用する．いちいち全体をコンパイルする必要のない利点を活用する．
- ヘルプ上の例は全てコピーできる．
- ctrl+shift+B で括弧の対応が分かる．

Module を使った簡単な関数の例：右・左側通行の Agent-based モデル

```
cogame2[n_,A_,B_,it_]:=Module[{rrvec,pright,partner,right,payoff},
pright=Table[RandomReal[],{i,1,n}];(* 初期値　右選択確率　*)
rrvec=Table[0,{it}];(* 初期値　右選択率のベクトル　*)
(*ここから Do ループを it 回繰り返す.　*)
Do[(* 繰り返しの度にマッチング　*)partner=Module[{x,p},
x=RandomSample[Range[n]];p=Range[n];
Table[p[[x[[i]]]]=x[[n+1-i]],{i,1,n}];p];
(* 戦略決定 *)
right=Table[If[pright[[i]]>RandomReal[],1,0],{i,1,n}];
(*利得決定*)payoff=Table[Which[
(right[[i]]==1)&&(right[[partner[[i]]]]==1),A,
(right[[i]]==1)&&(right[[partner[[i]]]]==0),0,
(right[[i]]==0)&&(right[[partner[[i]]]]==1),0,
(right[[i]]==0)&&(right[[partner[[i]]]]==0),B],{i,1,n}];
(*　学習 *)Table[Which[
((right[[i]]==1)&&(payoff[[i]]>0)),pright[[i]]=pright[[i]]+(A/(A+B)
(1-pright[[i]])),
((right[[i]]==0)&&(payoff[[i]]>0)),pright[[i]]=pright[[i]]-(B/(A+B)
(pright[[i]])),
payoff[[i]]==0,pright[[i]]=pright[[i]]],{i,1,n}];
rrvec[[j]]=(Plus@@right)/n,{j,1,it}];
rrvec]
```

10. ルーレットのシミュレーション

Q　初期値 1 万円を持ち，1 回に 1000 円を賭けるプログラムを If 文で考えよ．勝率は 1/2 とする．
Q　上のプログラムの 10 回繰り返しを考えよ．

ルーレットには投資額が二倍になる「赤 or 黒」という賭け方がある．初期所持金 m_0 が二倍になるまで，次のアルゴリズムで賭け続けると仮定する[3]（マルチンゲール法）．

1. 最初に p_0 円賭ける．もし勝ったら，同額を賭け続ける．
2. 負けた場合は次の回に現在の賭金の二倍を賭ける．一度勝ったら，また掛け金を p_0 に戻す．

倍賭法アルゴリズム（所持金が 2 倍になるまで繰り返すプログラム）のコード例．

```
rt1[m0_, p0_] := Module[{m = m0, n = 0, p = p0, a = {{0, m0}}},
  While[m > 0 && m < 2 m0,  n++;
   If[Random[] <= 9/19,
      m = m + p;  p = p0,
      m = m - p;  If[2 p > m, p = m, p = 2 p]];
      a = Append[a, {n, m}]   ];(*While*)
  ListPlot[a, PlotRange -> {{0, 200}, {0, 2 m0}}]
  ]
```

10.1. 変数の意味

m0　初期所持金	m　現在の所持金
p0　初期賭け金	p　現在の賭け金
n　繰り返し回数	a　結果のリスト｛回数，その回の金額｝

[3] このような倍賭方式をマルチンゲール法という．確率過程としてのマルチンゲールは $E(S_{n+1}|S_n,S_{n-1},...,S_1)=S_n$ という性質を指す．単純なランダムウォークは $p=q$（確率が対称）のときマルチンゲールとなる．

10.2. ソースの解説

While[m > 0 && m < 2 m0,実行文 1;実行文 2;実行文 3]

条件「(m > 0) && (m < 2 m0)」を満たすあいだ，実行文 1，2，3 を繰り返す．条件の意味は「現在の所持金 m が 0 より大きい」かつ「現在の所持金が初期所持金 m0 の二倍を超えていない」である．逆に言えば，「破産するか，目標金額に到達するまでは繰り返す」という意味である[4]．実行文は次の三つである．

実行文 1　n++;

回数 n に 1 を足す．n=n+1 と書いてもいい．++をインクリメントという．一巡するたびに数が増えるカウンタによく使う．

実行文 2　If[Random[] <= 9/19,

```
m = m + p; p = p0,
m = m - p;
If[2 p > m, p = m, p = 2 p]];
```

実行文 2 は If 文の中に If 文が含まれている．くわしく書けば

```
If[Random[] <= 9/19,（if 条件 1）
    m = m + p; p = p0,（if 条件 1 を満たす場合に実行）
    m = m - p; 二つ目の If 文　（if 条件 1 が満たされない場合に実行）
    ];
```

という構造になっている．

If 条件 1 は「[0,1]内の乱数が 9/19 以下である．」を意味する．9/19 はルーレットに勝つ確率なので，ルーレットの成功条件と考えればよい．ルーレットに勝った場合，次の命令を実行する．

```
m = m + p; p = p0
```

現在の所持金 m に賭け金 p を足す；現在の賭け金 p に p0 を代入する（現在の賭け金 p を p0 に戻す）．なお m = m + p は省略的に m += p と書いてもよい．

ルーレットに負けた場合，次の命令を実行する．

```
m = m -p; 二つ目の If 文
```

所持金 m から賭け金 p を引き，二つ目の if 文を実行する．二つ目の if 文の意味は，こうで

[4] 「条件を満たすまで繰り返す」ことは「条件を満たした時に停止する」ことに等しい．

ある.

```
If[2 p > m, p = m, p = 2 p]
```

現在の所持金が現在の賭け金の 2 倍よりも小さい場合は，全額賭ける．まだ余裕がある場合は現在の賭け金を 2 倍する.

実行文 3　`a = Append[a, {n, m}]`

三つ目の実行文で結果をリスト a に追加している．毎回結果が追加されるので，回数文の結果がリストには含まれる.

`ListPlot[a, PlotRange -> {{0, 200}, {0, 2 m0}}]`

最後に変数 a に代入された結果をリストプロットしている．出力が離散的なデータなので，結果は単なる `plot` ではなく `ListPlot` を使う点に注意する.

Q　条件（勝利確率，初期賭け金，初期所持金）を変えながら，結果をプロットしてみよう．そこからどんなことが読み取れるかを考えよ.

さて分析の結果，持ち金を二倍にするのはなかなか難しいことが分かる[5]．そこで次のような問題を考える.

問題．一般に設定金額 gm だけ儲かれば，終了する関数を定義せよ.

解答例.

```
rtgm[m0_, p0_, gm_] :=
Module[{m = m0, n = 0, p = p0, a = {{0, m0}}},
  While[m > 0 && m < m0 + gm,
   n++;  If[Random[] <= 9/19,
    m += p; p = p0,
    m -= p; If[2 p > m, p = m, p = 2 p]];
   a = Append[a, {n, m}]]; Last[a]  ]
```

[5] 「なかなか難しい」ことを統計的に示すには，シミュレーションの反復が必要である.

マークした部分に注目せよ. 新しい引数として gm が追加され, While 条件で「m < m0 + gm」の部分が修正されている.

問題. 勝った場合にのみ次回倍賭けを実行して，負けると初期賭け金に戻すアルゴリズムを表現してみよ. その結果を《負けたら倍》の場合と比較せよ.

問題. このギャンブルを 365 回（1 日 1 回 1 年間）繰り返した場合の結果を示せ.

問題. `rtgm[m0_, p0_, gm_]` を用いて「1 千万円の初期資産を元手に 1 日 1 万円だけギャンブルで稼ぎ続けることができるか」分析せよ.

問題. 倍賭け法アルゴリズムを R で書きなさい.

10.3. Manipulate の使い方.

Manipulate はパラメータの変化に伴う計算結果の変化を比較する際に便利な関数である. アプリケーション風のインターフェイスが瞬時に作成できる.

例.

```
Manipulate[
 Plot[a + b x + c x^2, {x, -5, 5}],
 {a, 1, 5},
 {b, 1, 5},
 {c, 1, 5}]
```

a,b,c がパラメータ. x が独立変数であると解釈するとよい. 一般的には

```
Manipulate[ 関数,
 {パラメータ1, min, max},  {パラメータ2, min, max},   {パラメータ3, min, max}]
```

という形式である. スライダの他に，ボタンやプルダウン，チェックボックス等が使える.

10.4. 計算結果の統計

ルーレットを例に，計算結果の統計について解説する．

```
rtgm[m0_, p0_, gm_] :=
Module[{m = m0, n = 0, p = p0, a = {{0, m0}}},
  While[m > 0 && m < m0 + gm,
   n++;  If[Random[] <= 9/19, m += p; p = p0,
    m -= p; If[2 p > m, p = m, p = 2 p]];
   a = Append[a, {n, m}]]; Last[a]  ]
```

この関数は最終的な output が，リスト a の最後の要素なので，{10,2000}や{5,0}のように{回数，金額}の組み合わせで表示される．したがって 100 回繰り返した場合の破産確率を求めたければ，

```
Table[rtgm[1000, 1, 10],{100}]
```

で 100 回分の結果を出力して，ここから破産したケース数を数える．

問題．破産確率をもとめよ．
ヒント．`Count, Case` などの関数を利用する．

問題．倍賭法が初めて成功する回数の平均値をシミュレーションによって求めよ．

問題．上の問題の期待値が幾何分布の平均と一致することを確認せよ．

定義（幾何分布）．確率 p で成功する試行を繰り返して，初めて成功した回数を表す確率変数を X とおく．その確率関数は

$$P(X=r)=(1-p)^{r-1}p$$

である．平均と分散は $E(X)=1/p,\ V(X)=(1-p)/p^2$ である．

問題．倍賭法の純利益の期待値を求めよ．

10.5. ランダムウォーク

1 回毎のルーレットの結果を確率変数 X_i で表す（i は $i=1,2,...,n$ である）．この確率変数は勝った時には 1，負けた時には -1 という値をとると仮定する．

勝つ確率を 1/2（一般的には p ），負ける確率を 1/2（一般的には $1-p=q$ ）とおけば，

$$P(X_i=1)=1/2=p$$
$$P(X_i=-1)=1/2=1-p=q$$

である．ここで実現値が 0 と 1 ではなく，−1 と 1 であることに注意する．

次に各 X_i が独立である，つまり毎回のルーレットの結果は，過去の結果からまったく影響をうけないと仮定する．すると，1 回目の結果から n 回目の結果の和は

$$S_n=X_1+X_2+\cdots+X_n$$

で表される．この確率変数 S_n を**ランダムウォーク**という．S_n の添え字 n は時刻を表していて，時刻 0 の位置は $S_0=0$ と定義しておく．

Q.　n 回のルーレットで $S_n=k$ となる確率は何か？

考え方 : 勝てば 1 進み，負ければ -1 進むような運動だと考える．ゆえに n 時間後に位置が k であるような確率を考えれよい．勝数の分布は二項分布に従うことが分かっているから，k 回勝つ確率を考えれば，位置が k である確率と一致するはずである．

まず，n 回の試行で k 回勝つ確率は二項分布に従うから

$$P(X=k)={}_nC_k\left(\frac{1}{2}\right)^k\left(\frac{1}{2}\right)^{n-k}$$

である．ただし n 回後に $S_n=k$ となる確率は

$$P(S_n=k)={}_nC_k\left(\frac{1}{2}\right)^k\left(\frac{1}{2}\right)^{n-k}$$

ではないことに注意する．なぜなら現在位置が k であるときに，勝利数も k に一致するとは限らないからである．このことを単純な例で考えてみる．

$n=5$ の場合を考え，勝利した数を 3，負けた数を 2 とおく．たとえば最初に 3 回勝って，残り 2 回を負けると，現在位置は

$$S_n=1+1+1-1-1=3-2=1$$

となる．この場合は，明らかに《勝利数 3》と《現在の位置 1》は異なる．なぜなら，ランダムウォークの軌道は，負けたときに−1 進むからである．

言い換えれば，勝利数と現在位置の関係は

現在位置＝勝利数―負け数

になっているはずである.

同じことだが

$$n = \text{勝利数} + \text{負け数}$$

でもある.

現在位置は《勝利数》と《負け数》によって決まるから，さきに《勝利数》と《負け数》を仮定しないと，現在位置 x を確率変数とみなした場合の確率を計算できない．ここで n は試行回数として与えられているから，実質的には《勝利数》と《負け数》のどちらかを決めれば，現在位置が決まることに注意する.

例えば $n=5$ の場合，勝利数を $w=3$，負け数を $n-w=2$ とおく． $S_n=1$ となる確率は《$n=5$ で勝利数が 3 になるような二項分布》の確率に等しいから･･･

$$P(S_n=1)=P(X=3)$$
$$={}_nC_w\left(\frac{1}{2}\right)^w\left(\frac{1}{2}\right)^{n-w}={}_5C_3\left(\frac{1}{2}\right)^3\left(\frac{1}{2}\right)^2$$

である．ただしこの計算は勝利数 w を先に決めておかないと確定できないので，現在位置から勝利数を逆算する方法を考える.

一般的な記号として勝利数を w，負け数を l とおく．現在位置 k は

$$\text{現在位置}\, k = w - l$$

である．また試行回数 n は勝利数 w と負け数 l の合計

$$\text{試行回数}\, n = w + l$$

である．二つの式を足すと

$$k+n=(w-l)+(w+l)$$
$$k+n=2w$$
$$w=\frac{k+n}{2}$$

になる．つまり勝利数 w は現在位置と試行回数 n で一意的に決まる.

この関係を二項分布の確率関数に代入すると

$$P(S_n=k)=P(X=w)={}_nC_w\left(\frac{1}{2}\right)^w\left(\frac{1}{2}\right)^{n-w}$$
$$={}_nC_{(k+n)/2}\left(\frac{1}{2}\right)^{w+n-w}={}_nC_{(k+n)/2}\left(\frac{1}{2}\right)^n$$

である．ただし k の範囲は，$-n$ から n までの整数とする.

命題（ランダムウォークの平均と分散）．ランダムウォークの n 期後の位置 S_n の平均 $E[S_n]$ と分散 $V[S_n]$ は

$$E[S_n] = n(p-q),\ \ V[S_n] = 4npq$$

である．

証明． $E[X_i] = p-q, V[X_i] = 4npq, (i = 1,2,...,n)$ より，各 X_i が独立であるという仮定から $E[S_n] = E[X_1] + E[X_2] + \cdots + E[X_n]$ である．ここで

$$E[X_1] = E[X_2] = \cdots = E[X_n] = p-q$$

だから，これを n 個足した $E[S_n]$ は

$$E[S_n] = n(p-q)$$

である．一方分散についても，各 X_i が独立であるという仮定から

$$\begin{aligned} V[S_n] &= V[X_1 + X_2 + \cdots X_n] \\ &= V[X_1] + V[X_2] + \cdots + V[X_n] \end{aligned}$$

である.ここで

$$V[X_i] = E[X_i^{\,2}] - E[X_i]^2$$

という関係をつかう（この関係は任意の確率変数について成立する）．

$$E[X_i^{\,2}] = 1^2 p + (-1)^2 q = p + q = 1$$

だから

$$\begin{aligned} V[X_i] &= E[X_i^{\,2}] - E[X_i]^2 = 1 - (p-q)^2 \\ &= (1 + (p-q))(1 - (p-q)) \\ &= (1 + p - q)(1 - p + q) = (2p)(2q) = 4pq \end{aligned}$$

したがって $V[S_n] = V[X_1] + V[X_2] + \cdots + V[X_n] = 4npq$ である．

10.6. 所得分布の生成モデル

多くの近代社会の所得分布は対数正規分布に近似的に従うことが知られている．ルーレットのシミュレーションが示すような単純な投資ゲームを社会にいる人々が繰り返した結果として，対数正規分布を導出できる．

命題（所得分布の生成）．1 回あたり確率 p で成功する投資ゲームを n 回反復したと仮定する．初期所得 y_0 を持ち，毎回所得の b 割合（$0 < b < 1$）をコストとして投資する．n 回反復後の所得分布は対数正規分布に従う．

証明．n 回ゲームを繰り返したとき，w 回成功して，$n-w$ 回失敗すると，トータルの利益は $y_n = y_0(1+b)^w(1-b)^{n-w}$ である．y_n の対数をとると，

$$\begin{aligned}\log y_n &= \log\{y_0(1+b)^w(1-b)^{n-w}\}\\ &= \log y_0 + \log(1+b)^w + \log(1-b)^{n-w}\\ &= \log y_0 + w\log(1+b) + (n-w)\log(1-b)\\ &= \log y_0 + w\log(1+b) + n\log(1-b) - w\log(1-b)\\ &= w\log\left(\frac{1+b}{1-b}\right) + \log y_0 + n\log(1-b)\end{aligned}$$

となる．

ベルヌイ試行を反復した場合の成功回数 w は二項分布にしたがう．また《ド・モアブル＝ラプラスの定理》によって，ゲーム回数 n が十分に大きいとき，w の分布を正規分布で近似できる．ここで，w の係数である

$$\log\left(\frac{1+b}{1-b}\right)$$

と，第 2 項と第 3 項の和

$$\log y_0 + n\log(1-b)$$

は確率変数ではなく，モデルの仮定によって与えられるパラメータ――つまり定数である．

定数の部分を A,B で表すと，

$$A = \log\left(\frac{1+b}{1-b}\right),\ B = \log y_0 + n\log(1-b)$$

と表すことができる．言い換えると総所得は

$$\log y_n = wA + B$$

と表すことができる．右辺は $n \to \infty$ のとき中心極限定理により正規分布に従うので，y_n は対数正規分布にしたがう（y_n の対数が正規分布に従うことが示された）．

□

問題．命題が正しいことを示す具体例をシミュレーションによって確かめよ．

命題(所得分布の確率密度関数). 反復投資ゲームから生成される所得分布の確率密度関数は

$$f(y)=\frac{1}{\sqrt{2\pi np(1-p)A^2}}\frac{1}{y}\exp\left\{-\frac{1}{2}\frac{(\log y-B-Anp)^2}{np(1-p)a^2}\right\}$$

$$\text{where } A=\log\left(\frac{1+b}{1-b}\right),\ B=\log y_0+n\log(1-b)$$

である.

証明. 1 回の投資は確率 p で成功するベルヌイ試行である. よって n 回反復した場合に, 成功数 W は 2 項分布に従う. n が十分に大きいとき, 2 項分布の確率関数は正規分布の確率密度関数によって近似できる（ド・モアブル=ラプラスの定理). 具体的には

$$P(W=k)={}_nC_k p^k(1-p)^{n-k}$$

を

$$f(w)=\frac{1}{\sqrt{2\pi np(1-p)}}\exp\left\{-\frac{(w-np)^2}{2np(1-p)}\right\}$$

で近似する. 初期所得 y_0 からスタートして, 毎回所得の b 割合（$0<b<1$）をコストとして投資する. n 回ゲームを繰り返したとき, w 回成功して, $n-w$ 回失敗すると, トータルの利益を y とおけば $y=y_0(1+b)^w(1-b)^{n-w}$ である. 対数変換すると

$$\log y=Aw+B$$

$$A=\log\left(\frac{1+b}{1-b}\right),B=\log y_0+n\log(1-b)$$

だった. $\log y=x$ とおけば, $x=Aw+B$ だから, x の確率密度関数は, 変数変換によって

$$f(x)=\frac{1}{\sqrt{2\pi np(1-p)}}\exp\left\{-\frac{(\frac{x-B}{A}-np)^2}{2np(1-p)}\right\}\cdot\frac{dw}{dx}$$

$$=\frac{1}{\sqrt{2\pi np(1-p)A^2}}\exp\left\{-\frac{1}{2}\frac{(w-B-Anp)^2}{np(1-p)a^2}\right\}$$

である. 最後に $\log y=x$ とおいて, 変数変換した確率密度関数が求める y の密度である.

$$f(y)=\frac{1}{\sqrt{2\pi np(1-p)A^2}}\frac{1}{y}\exp\left\{-\frac{1}{2}\frac{(\log y-B-Anp)^2}{np(1-p)a^2}\right\}$$

$$\text{where } A=\log\left(\frac{1+b}{1-b}\right),\ B=\log y_0+n\log(1-b)$$

命題（不平等が増大する条件）．投資ゲームの反復回数 n が大きいほど，また投資割合 b が大きいほど不平等は増大する．また p=0.5 のとき他の条件が一定であれば不平等が最大化する．

証明．Aitchison and Brown (1957)の定理より，対数正規分布 $\Lambda(\mu,\sigma^2)$ に従う所得分布のジニ係数は

$$G = 2\int_{-\infty}^{\frac{\sigma}{\sqrt{2}}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx - 1$$

である．ジニ係数は標準正規分布の分布関数で表すことができる．積分の上端が $\sigma/\sqrt{2}$ なので，パラメータ σ が増加するほど，ジニ係数は大きくなる．いま投資ゲームから導出した対数正規分布のパラメータ σ は

$$\sigma = np(1-p)\log\left(\frac{1+b}{1-b}\right)$$

だから，これをモデルのパラメータ n,b,p で微分することで，命題を得る．たとえば，

$$\frac{d}{db}\log\left(\frac{1+b}{1-b}\right) = \frac{1}{x}\cdot\left(\frac{1+b}{1-b}\right)' = \frac{1-b}{1+b}\frac{1(1-b)-(1+b)(-1)}{(1-b)^2}$$
$$= \frac{1}{1+b}\frac{1-b+1+b}{(1-b)} = \frac{2}{1-b^2} > 0$$

である．そのほかは，自明であろう．

11. 出会いの数理モデル

11.1. 2 項分布

ある期間中，n 人の異性とあなたは出会う．x 人の異性があなたを好きになる確率は 2 項分布の確率関数によって与えられる．仮定：

1. n 人の異性と出会う
2. 一人の異性は確率 p であなたを好きになる（$0 \le p \le 1$）.
3. 各異性は独立にあなたに対して好意を持つ.

以上の仮定を満たすとき，あなたが x 人から好かれる確率は

$$\Pr(X = x) = {}_nC_x p^x (1-p)^{n-x}$$

で決まる．ゆえに x 以上から好かれる確率 $\Pr(X \ge x)$ は

$$1 - \Pr(X < x) = 1 - \sum_{k=0}^{x-1} {}_nC_k p^k (1-p)^{n-k}$$

である.

問題．大学の 1 年間で 50 人の人と出会い，1 人あたり 5%の確率で好かれると仮定した場合に，あなたが 3 人以上から好かれる確率を計算しなさい.

ヒント.

```
mote[n_, p_, x_] := 1 - CDF[BinomialDistribution[n, p], x - 1]
```

問題．n や p を変化させて分析しなさい

問題．任意の x について，$\Pr(X \ge x)$ が p の増加関数であることを証明せよ.

問題．任意の x について，$\Pr(X \ge x)$ が n の増加関数であることを証明せよ.

問題．今後 20 年間であなたにモテ期が何回訪れるか予想しなさい.

11.2. インプリケーション

命題．n, x を所与とするとき，x 人から好かれる確率は $x > np$ の場合に，p に関して増加である．ただし $0 < p < 1$ を仮定する．

証明．

$$\frac{d}{dp}P(X = x) = {}_nC_x p^{x-1}(1-p)^{n-x-1}(x - pn)$$

より

$$\frac{d}{dp}P(X = x) = \begin{cases} \text{正}, & x > np \\ 0, & x = np \\ \text{負}, & x < np. \end{cases}$$

□

命題．任意の x について，$\Pr(X \geq x)$ は p の増加関数である．

証明．まず微分するために確率 $P(X \geq x)$ を具体的にかくと，

$$\begin{aligned} P(X \geq x) &= 1 - F(x-1) \\ &= 1 - \sum_{k=0}^{x-1} {}_nC_k p^k (1-p)^{n-k} \end{aligned}$$

である．なるべく簡単な形で考えるために

p が増加すると $F(x)$ が増加する ⇔ p が増加すると $1 - F(x)$ が減少する

と言い換える． $F(x)$ を具体的に書けば

$$F(x) = \sum_{k=0}^{x} \binom{n}{k} p^k (1-p)^{n-k} .$$

$x = 0$ とおいて $F(0)$ を p で微分すると

$$\begin{aligned} F(0) &= \sum_{k=0}^{0} \binom{n}{k} p^k (1-p)^{n-k} \\ &= \binom{n}{0} p^0 (1-p)^{n-0} = (1-p)^n \end{aligned}$$

ここで

$$f(p)=(1-p)^n, \qquad x=1-p \text{とおけば}$$
$$(1-p)^n=(x)^n$$

だから，合成関数の微分を使って

$$\frac{df}{dp}=\frac{df}{dx}\frac{dx}{dp}=n(x)^{n-1}\cdot(-1)=-n(1-p)^{n-1}.$$

これは明らかに$0<p<1$の範囲では負である．

x=1 の場合

$$F(1)=\sum_{k=0}^{1}{}_nC_k\,p^k(1-p)^{n-k}$$
$$={}_n\,C_0\,p^0(1-p)^{n-0}+{}_n\,C_1p^1(1-p)^{n-1}=(1-p)^n+np(1-p)^{n-1}$$

である．第 1 項は前の条件と同じだから，第 2 項だけ追加すればよい．つまり

$$f(p)=(1-p)^n+np(1-p)^{n-1}$$

とおいて $f(p)$ を p で微分すればよい． $np(1-p)^{n-1}$ の部分は

$$np\times(1-p)^{n-1}$$

という積の形になってるから

$$(f(x)g(x))'=f'(x)g(x)+f(x)g'(x)$$

という定理を使う．すると・・・・・

$$\frac{d}{dp}f(p)=-n(1-p)^{n-1}+\{n(1-p)^{n-1}+np(n-1)(1-p)^{n-2}\cdot(-1)\}$$
$$=-n(1-p)^{n-1}+\{n(1-p)^{n-2}((1-p)-p(n-1))\}$$
$$=-n(1-p)^{n-1}+\{n(1-p)^{n-2}(1-p-pn+p)\}$$
$$=-n(1-p)^{n-1}+n(1-p)^{n-2}(1-pn)$$
$$=n(1-p)^{n-2}\{-(1-p)+1-pn\}$$
$$=n(1-p)^{n-2}\,p(1-n)=-n(n-1)\,p(1-p)^{n-2}.$$

やはり $p=0,1$のときは 0 で$0<p<1$の範囲では負である．以上を続けると

$x=0$のとき　$\dfrac{dF(x)}{dp}=-n(1-p)^{n-1}$

$x=1$のとき　$\dfrac{dF(x)}{dp}=-n(n-1)\,p(1-p)^{n-2}$

$x=2$ のとき　$\dfrac{dF(x)}{dp}=-\dfrac{n(n-1)(n-2)}{2}(1-p)^{n-3}p^2$

$x=3$のとき　$\dfrac{dF(x)}{dp}=-\dfrac{1}{6}(n-3)(n-2)(n-1)np^3(1-p)^{n-4}$

$x=4$ のとき　$\dfrac{dF(x)}{dp}=-\dfrac{1}{24}(n-4)(n-3)(n-2)(n-1)(1-p)^{n-5}np^4$

となる．ここで規則性が見えやすいように，少し結果の表示を変えてみると

$x=0$ のとき　$\dfrac{dF(x)}{dp}=-\left(\dfrac{1}{0!}\right)\times n\times p^0(1-p)^{n-1}$

$x=1$ のとき　$\dfrac{dF(x)}{dp}=-\left(\dfrac{1}{1!}\right)\times n(n-1)\times p^1(1-p)^{n-2}$

$x=2$ のとき　$\dfrac{dF(x)}{dp}=-\left(\dfrac{1}{2!}\right)\times n(n-1)(n-2)\times p^2(1-p)^{n-3}$

$x=3$ のとき　$\dfrac{dF(x)}{dp}=-\left(\dfrac{1}{3!}\right)\times n(n-1)(n-2)(n-3)\times p^3(1-p)^{n-4}$

$x=4$ のとき　$\dfrac{dF(x)}{dp}=-\left(\dfrac{1}{4!}\right)\times n(n-1)(n-2)(n-3)(n-4)\times p^4(1-p)^{n-5}$

ここから，$x=k$ のとき，一般項の形は

$x=k$ のとき　$\dfrac{dF(x)}{dp}=-\left(\dfrac{1}{k!}\right)\times n(n-1)(n-2)\cdots(n-k)\times p^k(1-p)^{n-(k+1)}$

であると予想できる．階乗の部分をまとめると

$$\frac{dF(x)}{dp}=-\frac{n!}{k!(n-(k+1))!}p^k(1-p)^{n-(k+1)}$$

である．最後に，一般項が正しいことを数学的帰納法を使って確認する．つまり

任意の x について，$\dfrac{dF(x)}{dp}$ が

$$\frac{dF(x)}{dp}=-\frac{n!}{x!(n-(x+1))!}p^x(1-p)^{n-(x+1)}$$

であることを示す．

1)　$x=0$ のとき,

$$\frac{dF(0)}{dp}=-\frac{n!}{0!(n-(0+1))!}p^{0}(1-p)^{n-(0+1)}$$
$$=-\frac{n!}{(n-1)!}(1-p)^{n-1}=-n(1-p)^{n-1}$$

である．よって成立する.

2) $x=k$ のとき成立すると仮定して，$x=k+1$ のときにも成立することを示す.

まず仮定より

$$\frac{dF(k)}{dp}=-\frac{n!}{k!(n-(k+1))!}p^{k}(1-p)^{n-(k+1)}$$

である．つぎに

$$\frac{dF(k+1)}{dp}$$

を直接計算する.

ところで

$$F(k+1)=\sum_{i=0}^{k+1}{}_{n}C_{i}\,p^{i}(1-p)^{n-i}$$
$$=\sum_{i=0}^{k}{}_{n}C_{i}\,p^{i}(1-p)^{n-i}+{}_{n}C_{k+1}p^{k+1}(1-p)^{n-(k+1)}$$

だから，第 2 項を 2 項分布の確率関数 $f(k+1)$ として分けておく事ができる．したがって，$F(k+1)$ を p について微分するということは,

$$F'(k+1)=\frac{dF(k)}{dp}+\frac{df(k+1)}{dp}$$

を計算することに等しい．さきに

$$\frac{df(k+1)}{dp}$$

の部分を計算しておこう.

$$\frac{df(k+1)}{dp}={}_{n}C_{k+1}\{(k+1)p^{k}(1-p)^{n-(k+1)}-p^{k+1}(n-(k+1))(1-p)^{n-(k+1)-1}\}$$
$$={}_{n}C_{k+1}\{p^{k}(1-p)^{n-k-2}((k+1)(1-p)-p(n-(k+1))\}$$
$$={}_{n}C_{k+1}\{p^{k}(1-p)^{n-k-2}(k-pn+1)\}$$

である．あとはこれを組み合わせて

$$F'(k+1)=\frac{dF(k)}{dp}+\frac{df(k+1)}{dp}$$

を計算すればいい.

$$\frac{dF(k)}{dp}$$

の部分は，仮定

$$\frac{dF(k)}{dp}=-\frac{n!}{k!(n-(k+1))!}p^k(1-p)^{n-(k+1)}$$

が使えるから，

$$\begin{aligned}
F'(k+1)&=\frac{dF(k)}{dp}+\frac{df(k+1)}{dp}\\
&=-\frac{n!}{k!(n-(k+1))!}p^k(1-p)^{n-(k+1)}+\frac{n!}{(k+1)!(n-(k+1))!}\{p^k(1-p)^{n-k-2}(k-pn+1)\}\\
&=\frac{n!}{k!(n-(k+1))!}p^k(1-p)^{n-k-2}\left\{\frac{1}{k+1}(k-pn+1)-(1-p)\right\}\\
&=\frac{n!}{k!(n-(k+1))!}p^k(1-p)^{n-k-2}\left\{\frac{-p(n-k-1)}{k+1}\right\}\\
&=-\frac{n!}{(k+1)!(n-(k+1))!}p^{k+1}(1-p)^{n-k-2}(n-k-1)\\
&=-\frac{n!(n-k-1)}{(k+1)!(n-k-1)!}p^{k+1}(1-p)^{n-k-2}\\
&=-\frac{n!}{(k+1)!(n-k-2)!}p^{k+1}(1-p)^{n-k-2}
\end{aligned}$$

である．この式は確かに，命題が $x=k+1$ のときにも成立することを示している．よって数学的帰納法により，

任意の x について，$\frac{dF(x)}{dp}$ が

$$\frac{dF(x)}{dp}=-\frac{n!}{x!(n-(x+1))!}p^x(1-p)^{n-(x+1)}$$

が成立することが示せた.

□

命題. 任意の x について， $\Pr(X \geq x)$ は n の増加関数である．

証明. 《x 人より多くの人に好かれる確率》は， x 人以下の人に好かれることの余事象だから，

$$1-P(X \leq x)=1-\sum_{k=0}^{x}{}_{n}C_{k}\,p^{k}(1-p)^{n-k}\ .$$

分布関数 $F(x)$ を使うと，

$$1-P(X \leq x)=1-F(x)$$

と表すことができる．パラメータが n であることを明示的に書けば

$$F_{n}(x)=\sum_{k=0}^{x}{}_{n}C_{k}\,p^{k}(1-p)^{n-k}$$

となる．出会う人が 1 人増えて $n+1$ 人になった場合，その分布関数は

$$F_{n+1}(x)=\sum_{k=0}^{x}{}_{n+1}C_{k}\,p^{k}(1-p)^{n+1-k}\ .$$

F の添え字の n や $n+1$ が出会う総数を表している．この表記を使って，

《n 人と出会って x 人より多くの人に好かれる確率 $1-F_{n}(x)$》

よりも

《$n+1$ 人と出会って x 人より多くの人に好かれる確率 $1-F_{n+1}(x)$》

のほうが大きいことを示せばいい．

つまり

$$1-F_{n}(x)<1-F_{n+1}(x)$$

であることが言えればいい．不等式を変形すれば

$$\begin{aligned}-F_{n}(x)&<-F_{n+1}(x)\\ F_{n}(x)&>F_{n+1}(x)\\ F_{n}(x)-F_{n+1}(x)&>0\end{aligned}$$

と同じ意味だから最後の $F_{n}(x)-F_{n+1}(x)>0$ を示す問題である．

分布関数の差を直接計算してみると

$$
\begin{aligned}
F_n(x)-F_{n+1}(x) &= \sum_{k=0}^{x} {}_nC_k\, p^k q^{n-k} - \sum_{k=0}^{x} {}_{n+1}C_k\, p^k q^{n+1-k} \\
&= \sum_{k=0}^{x} \frac{n!}{k!(n-k)!} p^k q^{n-k} - \sum_{k=0}^{x} \frac{(n+1)!}{k!(n+1-k)!} p^k q^{n+1-k} \\
&= \sum_{k=0}^{x} \frac{n!}{k!(n-k)!} p^k q^{n-k} - \sum_{k=0}^{x} \frac{(n+1)n!}{k!(n+1-k)(n-k)!} p^k q^{n-k} q \\
&= \sum_{k=0}^{x} \frac{n!}{k!(n-k)!} p^k q^{n-k} \left(1-\frac{(n+1)q}{n+1-k}\right)
\end{aligned}
$$

であり，正か負かは，簡単には判別できない．ただし

$$
\frac{n!}{k!(n-k)!} p^k q^{n-k}
$$

の部分は必ず正である． つまり，各項の正負は後ろの$\left(1-\frac{(n+1)q}{n+1-k}\right)$の部分で決まる．

$$
\begin{aligned}
\left(1-\frac{(n+1)q}{n+1-k}\right) > 0 \Leftrightarrow -\frac{(n+1)q}{n+1-k} > -1 \Leftrightarrow \\
\frac{(n+1)q}{n+1-k} < 1 \Leftrightarrow (n+1)q < n+1-k \Leftrightarrow \\
k+(n+1)q < n+1 \Leftrightarrow k < n+1-(n+1)q \Leftrightarrow \\
k < (n+1)(1-q) \Leftrightarrow k < (n+1)p
\end{aligned}
$$

$k < (n+1)p$ のとき，注目した項$\left(1-\frac{(n+1)q}{n+1-k}\right)$は正になっている．逆に言えば，$k \geq (n+1)p$ の場合は正ではない．これは，k がある程度大きくなってくると，$\left(1-\frac{(n+1)q}{n+1-k}\right)$が負になることを意味する．

確率関数をそれぞれ $f_n(x), f_{n+1}(x)$ とおいて総和の各項をまとめると

$$
\begin{aligned}
F_n(x)-F_{n+1}(x) &= \sum_{k=0}^{x} f_n(k) - \sum_{k=0}^{x} f_{n+1}(k) \\
&= \{f_n(0)-f_{n+1}(0)\} + \{f_n(1)-f_{n+1}(1)\} + \cdots + \{f_n(x)-f_{n+1}(x)\}
\end{aligned}
$$

である．各項は，$k < (n+1)p$ のとき正になっている．ここで

$$
\left(1-\frac{(n+1)q}{n+1-k}\right)
$$

は k に関して単調に減少するから，$k > (n+1)p$ を満たす第 k 番目の項は常に負になっている．つまり，$k > (n+1)p$ の場合，

$$\underbrace{\{f_n(0)-f_{n+1}(0)\}}_{\text{正}}+\underbrace{\{f_n(1)-f_{n+1}(1)\}}_{\text{正}}+\cdots+\underbrace{\{f_n(k)-f_{n+1}(k)\}}_{\text{負}}+\underbrace{\{f_n(k+1)-f_{n+1}(k+1)\}}_{\text{負}}+$$
$$\cdots+\underbrace{\{f_n(x)-f_{n+1}(x)\}}_{\text{負}}$$

である．ところで $f_n(x)$ に関しては， x は最大で n に等しい．その場合は

$$\begin{aligned}F_n(n)-F_{n+1}(n)&=1-F_{n+1}(n)=1-\{1-f_{n+1}(n+1)\}\\&=1-\{1-{}_{n+1}C_{n+1}p^{n+1}q^0\}=1-(1-p^{n+1})\\&=p^{n+1}>0\end{aligned}$$

が成立している．

$k<(n+1)p$ の場合，

$$F_n(k)-F_{n+1}(k)>0$$

であり， $k=n$ の場合も

$$F_n(n)-F_{n+1}(n)>0$$

である．

一方 $(n+1)p\le k\le n$ の範囲では，総和の各項は負になっている．つまり，それまでに足してきた和を超えるくらい大きな負の項がある場合には，全体として負になる可能性があると考えられる．

背理法を使って，そのような可能性がないことを示す．背理法の仮定として，

$(n+1)p\le k\le n$ の範囲で

$$F_n(k)-F_{n+1}(k)\le 0$$

であると仮定する．

$(n+1)p\le k\le n$ の範囲では確率関数の差

$$f_n(k)-f_{n+1}(k)$$

は負になっている．ところが，もし $(n+1)p\le k\le n$ の範囲にある，ある k で

$$F_n(k)-F_{n+1}(k)\le 0$$

になったとしたら，それ以降の項

$$\{f_n(k+1)-f_{n+1}(k+1)\},\{f_n(k+2)-f_{n+1}(k+2)\},\ldots,\{f_n(n)-f_{n+1}(n)\}$$

も負のはずだから， $F_n(n)-F_{n+1}(n)\le 0$ でなければならない．ところがこれは，先ほど示した

$$F_n(n)-F_{n+1}(n)>0$$

であるという事実に矛盾する．

よって背理法の仮定 $F_n(k)-F_{n+1}(k)\le 0$ は誤りで， $F_n(k)-F_{n+1}(k)>0$ が正しい．ゆえに

$$F_n(x) > F_{n+1}(x)$$

より

$$F_n(x) > F_{n+1}(x) \Rightarrow 1 - F_n(x) < 1 - F_{n+1}(x)$$

が成立する. つまり他の条件が等しければn人と出会う場合よりも$n+1$人と出会う方が，x人より多くの人に好かれる確率が大きい.

□

11.3. ベータ 2 項分布

一般に，人には好み(taste)の違いがあるため，好かれる確率 p は，なんらかの確率分布を持つと考えたほうが自然である．そこでつぎのようなモデルの精緻化を考える.

仮定．2 項分布のパラメータ p はベータ分布 $B(a,b)$ に従う確率変数である.

p は《確率》なので，実現値が[0,1]の範囲内にある確率変数だと解釈する.

定義（ベータ分布）．パラメータ a,b を持つベータ分布の確率密度関数は

$$f(p) = \frac{1}{B(a,b)} p^{a-1}(1-p)^{b-1}, \qquad (0 \le p \le 1)$$

である．ここに $B(a,b)$ はベータ関数と呼ばれる関数で,

$$B(a,b) = \int_0^1 t^{a-1}(1-t)^{b-1}\, dt$$

と定義される．一方，ガンマ関数と呼ばれる関数は

$$\Gamma(a) = \int_0^\infty t^{a-1}e^{-t}\, dt$$

と定義される．ベータ関数とガンマ関数とのあいだには

$$B(a,b) = \Gamma(a)\Gamma(b) / \Gamma(a+b) = B(b,a)$$

という関係がある.

問題．ベータ関数の確率密度関数のグラフを描け.

ヒント．BetaDistribution をヘルプで調べてみよう.

解答例．内蔵関数 Manipulate を使うと，パラメータの変化をスライダで比較できるので便利である．

問題．《ベータ 2 項分布》の確率関数のグラフを描け．
ヒント．BetaBinomialDistribution をヘルプで調べてみよう．

命題．x 人から好かれる確率はベータ 2 項分布の確率関数により与えられる．自分を好きになる人の総数の期待値は $E[X]=na/(a+b)$ である．ここで a,b はベータ分布のパラメータである．

証明．《何人から好かれるか》を確率変数 X （2 項分布に従う），ベータ分布に従う母数 P を確率変数 P とおく． X と P は確率変数として独立と考えられるので， x 人から好かれる確率は， X,p の同時確率関数の周辺確率分布によって与えられる．つまり

$$f(x)=\int_0^1 {}_nC_x p^x(1-p)^{n-x}\cdot\frac{1}{B(a,b)}p^{a-1}(1-p)^{b-1}dp$$

で与えられる．これを計算すると

$$\begin{aligned} f(x) &= \frac{{}_nC_x}{B(a,b)}\int_0^1 p^x(1-p)^{n-x}\cdot p^{a-1}(1-p)^{b-1}dp \\ &= \frac{{}_nC_x}{B(a,b)}\int_0^1 p^{x+a-1}(1-p)^{n-x+b-1}dp \\ &= {}_nC_x\frac{B(a+x,b+n-x)}{B(a,b)} \end{aligned}$$

である．

問題．50 人と出会い，母数 p が $B(10,10)$ に従う場合に，x 人から好かれる確率をベータ 2 項分布の確率関数を使って計算しなさい．

問題．母数 p が $B(a,b)$ に従う場合に期待値が $E[X]=na/(a+b)$ となることを証明しなさい．

11.4. 相思相愛の確率

出会いのモデルをベースに男女が相思相愛になる確率を考える．好意の双方向性を数学的に表現するために，自分が相手を好きになる事象を A とおいて，相手が自分を好きになる事象を B とおく．すると，お互いに好きになる事象は $A \cap B$ で表すことができる．《A かつ B》は，お互いに好きになった状態である．

ここで《自分が相手に好かれると，相手のことを好きになりやすい》という傾向を表現するために

自分が相手を好きになるという事象 A が生じる確率

と

相手が自分を好きだ（事象 B が成立）という条件の下で，自分が相手を好きになる確率

は違うと考える．つまり

自分が相手を好きになる確率は $P(A)$

と

相手が自分を好きだという条件の下で，自分が相手を好きになる確率 $P(A \mid B)$

を比較すると，

$$P(A) < P(A \mid B)$$

になるはずだ，と仮定する．

いま無条件に《好きになる確率》をそれぞれ

$$P(A) = p,\ P(B) = p$$

だと仮定すると．なんらかの正の確率 $\varepsilon > 0$ が存在して

$$P(\text{自分が相手を好き} \mid \text{相手が自分を好き}) = P(A \mid B) = p + \varepsilon$$

だと考えれば，$P(A) < P(A \mid B)$ となるから，自分が相手に好かれたときは，その相手を好きになりやすくなる傾向を表現できる．

条件付き確率の定義から

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

だから，これを変形すれば

$$P(A \mid B)P(B) = P(A \cap B)$$

となる．

$P(A \cap B)$ は《自分が相手を好き》で，かつ《相手も自分を好き》になっている確率だから，結局自分と相手が相思相愛になる確率は

$$P(A\cap B)=P(A|B)P(B)=(p+\varepsilon)p$$

となる．$\varepsilon=0$ の場合は，　$P(A\cap B)=p(p+\varepsilon)=p^2$ となる．このことは

$$P(A)=p,\ \ P(A|B)=p$$

を意味するから，自分が相手を好きになるかどうかは，相手が自分を好きかどうかとは独立だという意味になる（つまり，パラメータ ε の値によって，行為が独立な場合と独立でない場合の両方を表現できる）

11.5. 社会全体で相思相愛の組が x 組できる確率

男女の集合をそれぞれ

$$N_1=\{1,2,3,...,n\},\ \ N_2=\{1,2,3,...,n\}$$

で区別する．社会全体で発生する相思相愛の組を確率変数 X で表す．n 人の各男女について可能な相思相愛の組を ij で表すと

11,　12,　13,…, 1n,

21,　22,　23,..., 2n,

…,

n1,　n2,　n3,…, nn

となり，その組み合わせ総数は $n\times n=n^2$ である．

各組は全て発生確率が $p(p+\varepsilon)$，発生しない確率が $(1-p(p+\varepsilon))$ なのでベルヌイ試行と見なせる．最大 n^2 個の組から，x 個のカップルが実現する組み合わせの総数は ${}_{n^2}C_x$ である．

また x 組の相思相愛カップルが発生して，n^2-x 組の相思相愛カップルが発生しない事象の確率は $(p(p+\varepsilon))^x(1-p(p+\varepsilon))^{n^2-x}$ である．ゆえに，社会全体での相思相愛組数の実現数を確率変数 X で表すと，その確率関数は

$$P(X=x)={}_{n^2}C_x(p(p+\varepsilon))^x(1-p(p+\varepsilon))^{n^2-x}$$

である．確率分布は 2 項分布だから期待値は

$$E(X)=n^2p(p+\varepsilon),$$

分散は

$$V(X)=n^2p(p+\varepsilon)(1-p(p+\varepsilon))$$

である．

11.1. 浮気は許されない？

ただし，ここで次の点に注意する．例えば，X=3 のとき，

11, 12, 13,…, 1*n*,
21, 22, 23,..., 2*n*,
…,
*n*1, *n*2, *n*3,…, *nn*

という組み合わせができたと仮定する（左の数字は男性の番号，右の数字は女性の番号を示す）．この場合，《3 番の女性》は，《1 番の男性》と《2 番の男性》と，同時に両思いになってしまい，俗にいう二股が発生する．つまり前節の計算は，両思いになるカップルの規範的に妥当な数ではなく，可能な両思いの組数を与えているに過ぎない（二股が発生しない組み合わせとなると，実際にはもっと少なくなる）．逆に言えば，この計算から潜在的な浮気の発生確率がわかる．

Q. 自分が n 人の異性と出会い，2 人以上の異性と両思いになる確率（つまり二股や三股をかけてしまう確率）を求めよ．

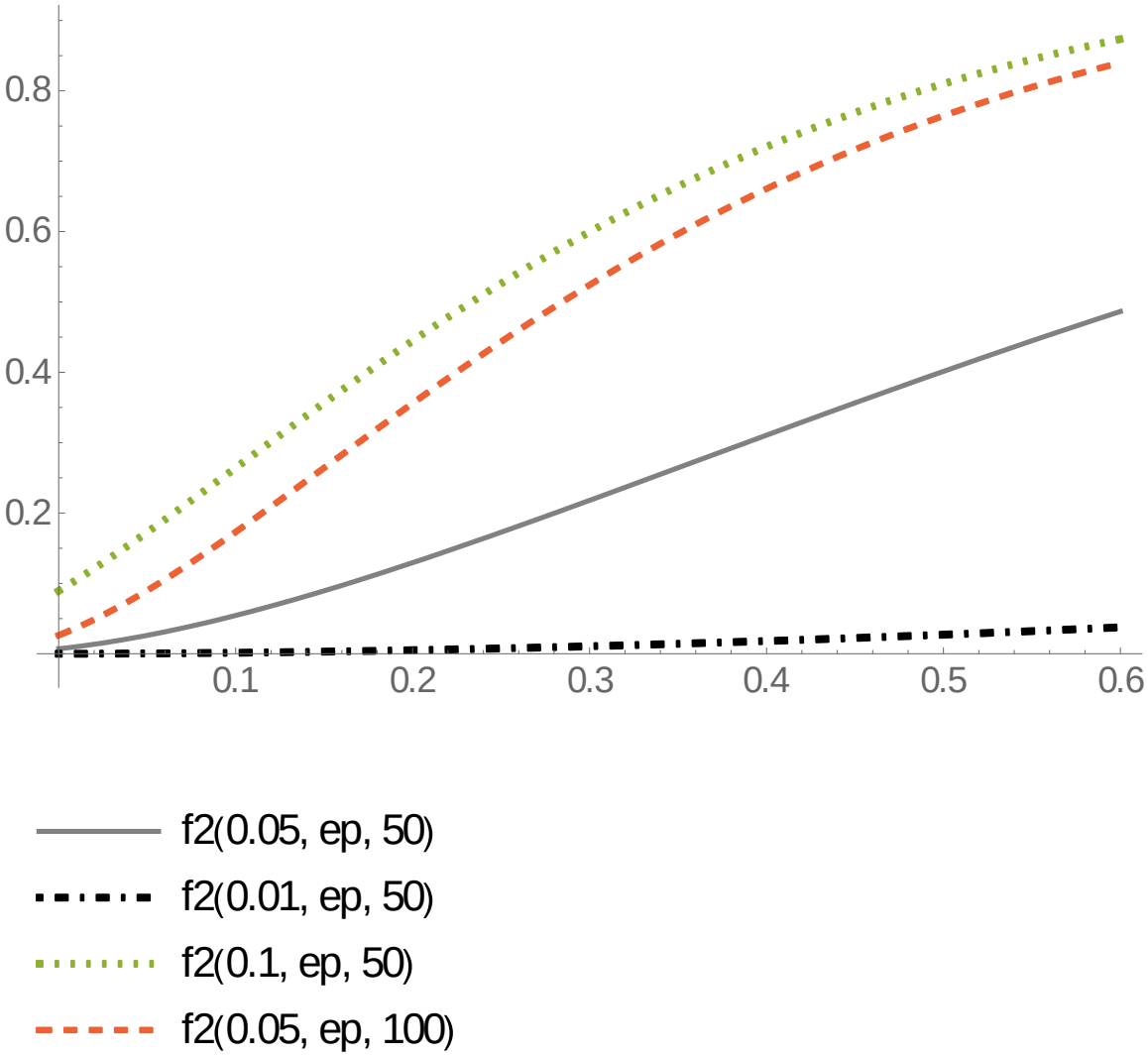
A. ふたたび個人の視点から考える．自分が x 人から好かれる確率は既に分かっているので，自分が x 人（$x \geq 2$）から好かれているという条件で，その中から 2 人以上に対して好意を持つ確率を考えればよい．

相手に好かれている条件の下で，相手のことを好きなる確率が $p+\varepsilon$ で与えられている事に注意すれば，2 人以上の異性と両思いになる確率は

$$\sum_{y=2}^{n} {}_nC_y p^y (1-p)^{n-y} \left\{ \sum_{x=2}^{y} {}_yC_x (p+\varepsilon)^x (1-p-\varepsilon)^{y-x} \right\}$$

である．

分析例を示そう．



図：パラメータは $n = 50,100; p = 0.01,0.05,0.1,$ として，横軸に ε をとり 0 から 0.6 まで変化させた．縦軸は確率である．

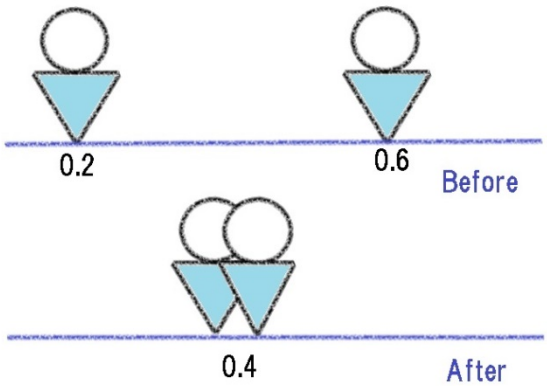
12． 中間階層の勃興と恋愛結婚の普及――恋愛の変容Ⅱ[6]

日本に精神性を重んじる西欧的な恋愛（ロマンティックラブ）が誕生したのは，明治以降だと言われる[7]. 1930 年頃に始まったとされる見合い結婚の減少と恋愛結婚の増加は, 2000 年頃には見合い結婚が 6.2%，恋愛結婚が 87.2%に達している（共に分母は婚姻数）．両者の割合は 1960 年頃に逆転しているという．結婚における恋愛結婚率が上昇すると恋愛がうまく出来ない個体は再生産できずに消滅する可能性が高くなる.

以下に自由恋愛に基づく結婚が中間階層の勃興と共に普及することをシミュレーションで示す.

Assumptions

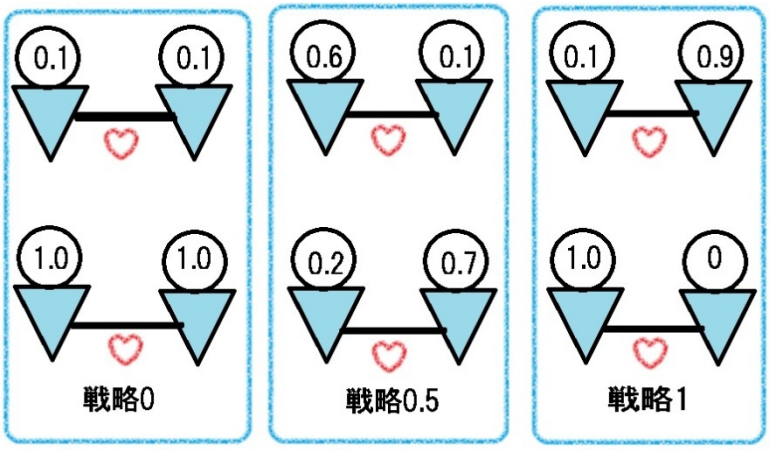
1. Random pairs of players are chosen, and they will marriage when the following conditions are met. When the pair succeeds in getting marriage, both of agents will receive benefit v>0. If they do not succeed, both individuals receive zero.
2. Each player has a class score, s, which is known to every other player. If a player marriage other player, then their class scores will be an average of their class scores. For example when a player with score 0 got marriage other player with score 1, their class scores are (0+1)/2=0.5. If they fail to marriage, the class score does not change.



図．結婚は，お互いの階層を平均化する作用を持つ

[6]浜田宏. 1997.「近代日本における恋愛の変容Ⅰ」『年報社会学論集』10:120-132.
[7]北村透谷の恋愛論などに《恋愛》という翻訳語が普及する以前は，男女間の性愛関係は単に恋と呼ばれていた．その特殊な洗練型として《粋》というコミュニケーション様式が江戸期に普及していたと言われている.

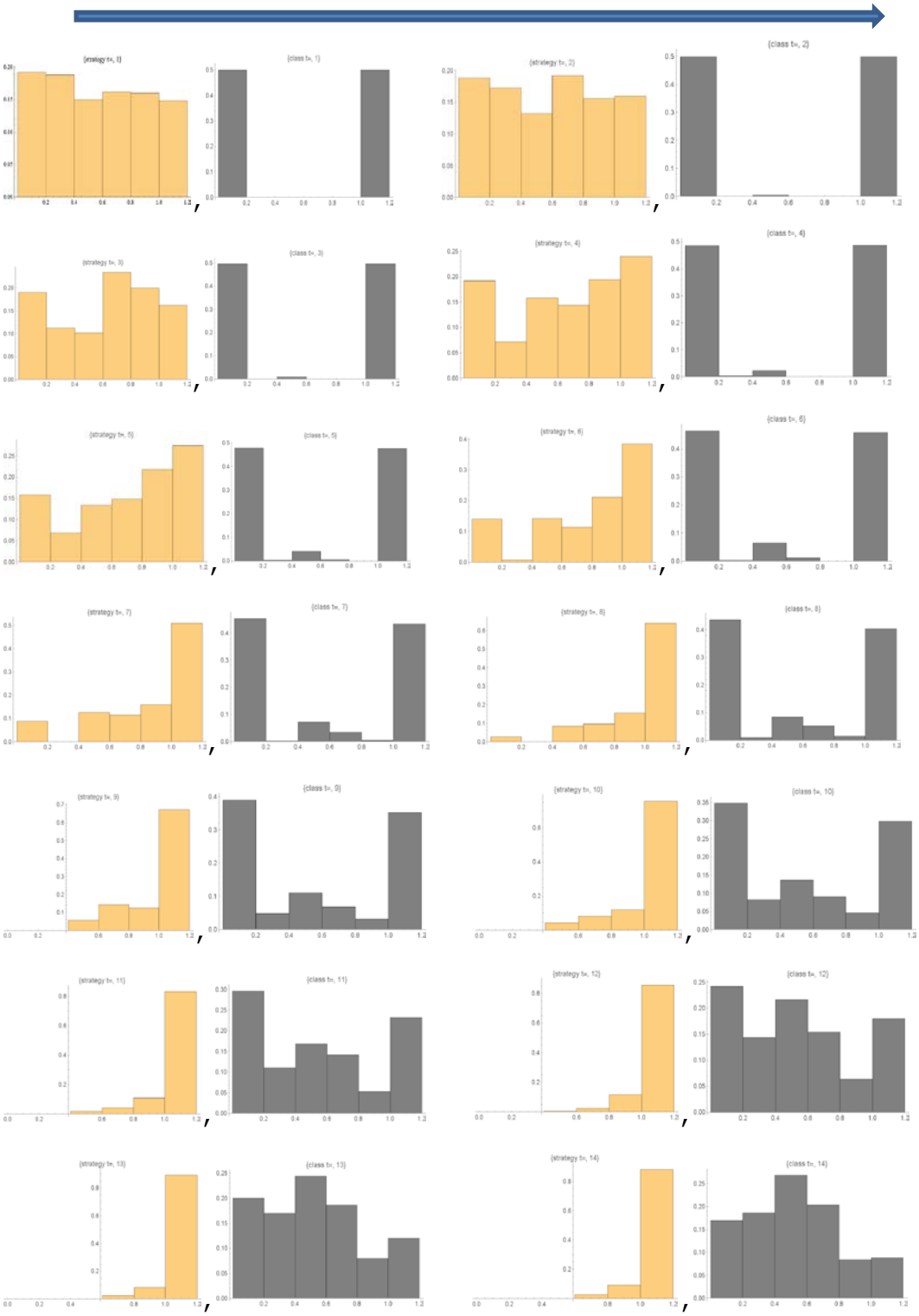
3. We consider strategies where players decide to marriage according to the class score of the partner. A strategy is given by a number x in {0, 0.2, 0.4, 0.6, 0.8, 1}. For any randomly matched players, they marriage if, and only if, the difference between class scores of the two players is smaller than their strategy x respectively. The strategy x=0 represents the players who want within class marriage, whereas the strategy x=1 represents the players who want class free marriage. At the beginning of generation, the class score are only low (score 0) and high (score 1) class. There is no middle class.



図．戦略によって結婚できる階層が異なる．戦略は 1 は誰とでも結婚できるが，戦略 0.5 は階層間距離が 0.5 以内の相手としか結婚できない．

4. In succession, m pairs are chosen. The fitness of a player is given by the total number of points received during the m interactions. Some players may never be chosen, in which case their payoff from the game will be 0.001. On average, a player will be chosen 2 m/n times.
5. At the end of each generation, players leave offspring in proportion to their fitness.

計算コードは少し長くなるので省略する（tr2.nb 参照）.
戦略 0 割合＝1/6,　高階層初期割合＝0.5,　集団人数＝500,　1 世代内のペア数＝100
世代数＝30,　婚成立時の利得 v＝1,
という条件下での計算結果を次の図に示す.



図．世代を経る毎に 2 極化していた階層から中間層が誕生と肥大化が進行しつつ，異類婚（≈恋愛結婚）のシェアが支配的になっていく様子が分かる．

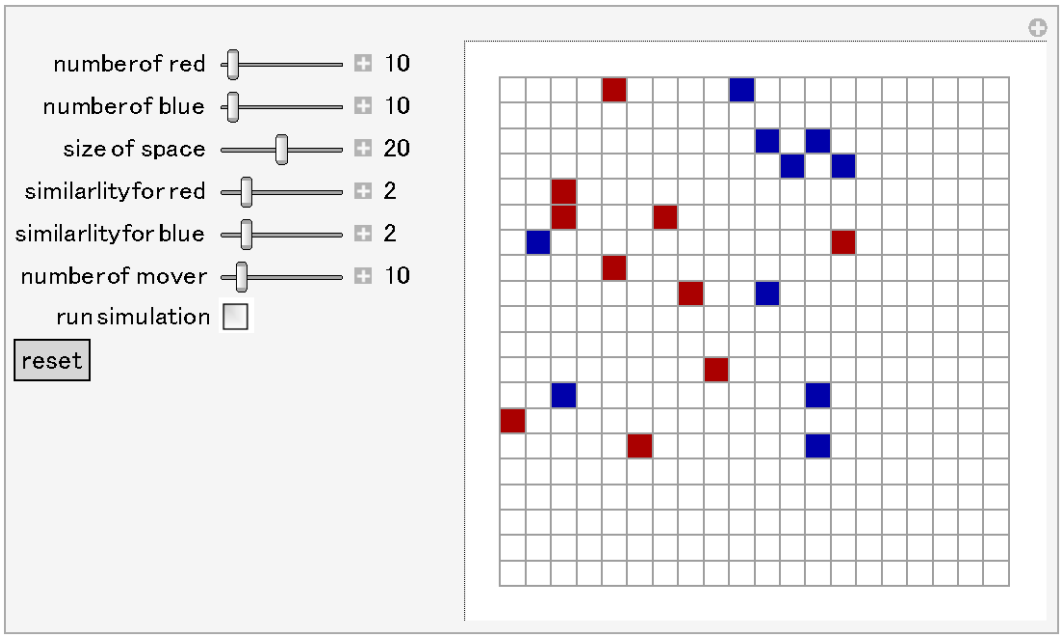
Q　このシミュレーションを解析的なモデルとして定式化できるか？

13.　セルラ・オートマトン[8]　　seg.nb

セルラ・オートマトンの例として，セグリゲーションのモデルと，その応用である避難行動のモデルを紹介する．トーマス・シェリングは，人種間で居住地分離が生じる仕組みを Agent based simulation によって個人的選択の集積として定式化した．世界中の国々に，外国人が集まって形成する街が存在する（例えばチャイナタウン）．このような人種に基づく自然発生的な分居は，次のようなモデルによって表現・説明できる（ただし以下の仮定は簡略化のためにオリジナルとは異なるので注意する）．

1. 社会には 2 種類（以上）の個人が存在する
2. 各個人は近隣のうち何人かは自分と同じ集団の成員であることを好む
3. 近隣にいる同種の個人数が許容水準を下回るとき，現在の自分の位置を移動する

モデルは個人を 2 次元格子上に配置して，その「近傍」を周りを取り囲む 8 個のセルで定義する（ムーア近傍）．近傍内の同種のエージェント数が設定値を下回る場合は，移動して，上回る場合は移動を続ける．



[8] Mathematica の開発者であるスティーブン・ウルフラムはセルラ・オートマトンの研究者でもあった．

この規模のプログラムになると複数のユーザー定義関数を組み合わせて作る必要がある．seg.nb 内の関数を実行して，それぞれの機能を確かめよう．

以下はグラフィカルな出力を得るための関数 ArrayPlot の練習例である．

Q. ArrayPlot を使って，3×3 マスの平面上に十字の形をプロットしなさい．

Q. RandomInteger を使って， 0 と 1 をランダムな要素として持つ，3×3 の行列を定義しなさい

Q. ArrayPlot を使って，3×3 マスの平面上にランダムな点を配置しなさい．

Q. ArrayPlot と Dynamic と Manipulate を組み合わせて，3×3 マスの平面上でランダムに点を動かしなさい．

参考例．グレースケールの点をランダムに動かすためのコード

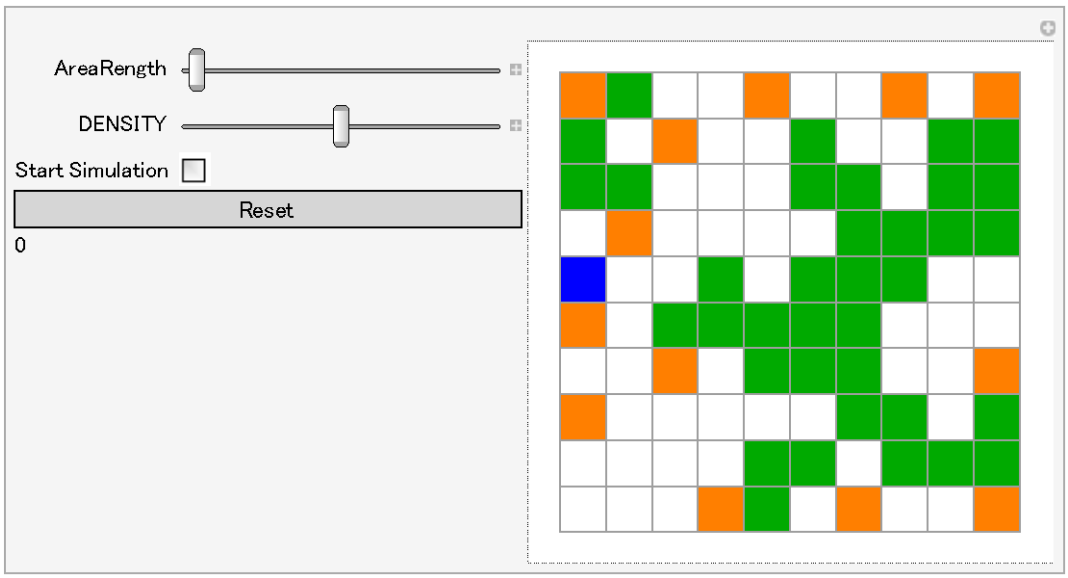
```
Manipulate[
 Dynamic[  Refresh[
   ArrayPlot[RandomReal[1, {20, 20}]],
   UpdateInterval -> If[moving, 0, Infinity]]
  ], {{moving, False, "Start"}, {True, False}}]
```

Q. ArrayPlot と Dynamic と Manipulate を組み合わせて，10×10 マスの平面上でランダムに『赤』と『青』の点を動かしなさい．

13.1. 避難行動のシミュレーション

セルラ・オートマトンの応用として，避難行動のシミュレーションを紹介する．仮定は次の通りである．

1. 各エージェントは指定された出口に向かい，ユークリッド距離が近いセルを確率的に移動する
2. 各セル上には 1 人のエージェントしか配置されない．
3. 1 セルに 2 人以上が同時に入った場合は，協力的エージェントのみが道を譲る．



コード作成：西塔貴海（2012 年度（2013 年 3 月）卒業）

セルラ・オートマトンは，視覚的なインパクトが強く，観る人を魅了するシミュレーションである．ただし「見た目が派手で凄そうにみえること」と「モデルとして説得力があること」は必ずしも一致しないので，注意が必要である（シェリングよりも複雑なモデルを作る事は簡単だが，モデルとして洗練させる事は難しい）．

おもしろいシミュレーションを作るための参考例として，web 上に公開されている Mathematica プログラムをダウンロードして動かしてみるとよい（ウルフラムリサーチ→デモンストレーションプログラムを検索）

ウルフラム自身による Mathematica プロジェクトの解説
http://www.ted.com/talks/lang/ja/stephen_wolfram_computing_a_theory_of_everything.html

14． 間接互恵性のモデル（情けは人のためならず）

このシミュレーションは

Nowak, M, A and Sigmund, K (1998) “Evolution of Indirect Reciprocity by Image Scoring.” *Nature* 393: 573-577.
にもとづく．

Assumptions

1. Random pairs of players are chosen, of which one is the potential donor of some altruistic act and the other is the recipient. The donor can cooperate and help the recipient at a cost *c* to himself, in which case the recipient receives a benefit of value b (with $b > c$). If the donor decides not to help, both individuals receive zero payoff.

集団の中からランダムにペアが抽出される．二人のうち，一人はドナーで，一人は受け手（recipient）である．ドナーはコストcを払って，受け手に協力できる．ドナーが協力した場合受け手は利得bをえる($b>c$)．ドナーが協力しない場合双方利得ゼロである．

2. Each player has an image score, s, which is known to every other player. If a player is chosen as a donor and decides to cooperate, then his (or her) image score increases by one unit; if the donor does not cooperate then it decreases by one unit. The image score of a recipient does not change.

各プレイヤーはイメージスコアsをもつ．この値は全員が知る．協力行動をとったドナーのスコアは1増加して．協力しなかった場合は1減少する．受け手のスコアは変化しない．

3. We consider strategies where donors decide to help according to the image score of the recipient. A strategy is given by a number *k*: a player with this strategy provides help if, and only if, the image score of the potential recipient is at least k. the strategy (*k*) values from −5 to +6. The strategy $k = -5$ represents unconditional cooperators, whereas the strategy *k*=6 represents defectors. The strategies are given by k_i and the image levels by s_i. A donor, *i*, cooperates with a recipient, *j*, if $k_i \le s_j$. At the beginning of each generation, the image levels of all players are zero.

ドナーがレシピエントのイメージスコアに従って協力するかどうかを戦略と考える．戦略kには無条件協力（−5）から常時裏切り（+6）までの幅がある．相手のイメージスコアが自分の戦略k以上のとき，ドナーは協力する．初期段階でイメージスコアはゼロである．

4. In succession, *m* donor-recipient pairs are chosen. The fitness of a player is given by the total number of points received during the *m* interactions. Some players may never be chosen, in which case their payoff from the game will be zero. On average, a player will be chosen 2 m/n times, either as donor or as recipient.

引き続いて，*m*個の「ドナー&受け手のペア」が選ばれる．プレイヤーの適合度は*m*回のやりとりで受け取ったポイント総数で決まる．プレイヤーの中には全く選ばれない者も存在し，その者の利得はゼロである．プレイヤー平均的に$2m/n$回，ドナーもしくは受け手として選ばれる．

5. At the end of each generation, players leave offspring in proportion to their fitness.

各世代の最後にプレイヤーは適合度に応じた子孫を残す.

このモデルは直接互恵ではなく，間接互恵によっても協力行動が進化することを示したシミュレーションである．とても単純な仮定だが，ダイナミックに更新される評判と協力の連動を的確に表現しており，協力行動に関するシミュレーションの最高傑作と呼べるモデルだろう．

コードの流れは《マッチング》《イメージスコアに基づく相互行為》《スコアと利得の更新》《適合度に基づく子孫の生産》である．

Q. *n* 人のプレイヤーから，*m* 個のペアを作る関数を定義しなさい．ただし同じ人が何回選ばれてもよい，と仮定する．

Q. マッチングした *m* 個のペアに関して，戦略 k とイメージスコアに基づいて行動を決定するプロセスを表現しなさい.
ヒント：必要な変数として，「各プレイヤーの戦略リスト」「各プレイヤーのイメージスコアリスト」「各プレイヤーの利得リスト」がある．

Q. ルーレットセレクションのために関数`RandomChoice`を使ったルーレットを作りなさい.

このモデルの Mathematica による再現は例えば次のようなコードによってなされる.

```
Module[{n = 100(*集団人数*), m = 200(*ペア個数*), gene = 150(*世代*),
  b = 1, c = 0.1, id(*n 人分の個体番号*), klist(*  n 人分の戦略*),
  slist(*n 人分の image score*),
  benefit(*n 人分の利得*), pair(*m 組の番号対*), fit(*k 個の適合度*)},

 (*初期条件の入力 *)
 id = Range[n];(* マッチング用個体番号 *)
 slist = Table[0, {n}];(*イメージスコア初期値 Print[slist];*)
 klist = Table[   Random[Integer, {-5, 6}], {n}];(*  戦略 k の初期値  Print["klist
初期値=",klist];*)
 benefit = Table[0, {n}];(*利得の初期値態 *)
Do[(* ここから 1 世代内での行動を指定する *)
  (*最初にイメージスコア slist，利得 benefit，適合度 fit を初期化する*)
  slist = Table[0, {n}];(* slist 初期化 *)
  benefit = Table[0, {n}];(* 利得 benefit 初期化 *)
  fit = Table[0, {12}];(*戦略別適合度 fit の初期化*)

  Do[(*「pair を一個だけ作る操作」を Do で m 回反復して m 個のペアを作る.
   slist はペアを 1 個作るたびに即時更新される．slist は 1 世代内(gene 内)でも更新される *)
   pair = RandomSample[id, 2];(*  ペアを一つ作る  *)

   If[klist〚pair〚1〛〛 <= slist〚pair〚2〛〛,(* 相手のイメージスコアをチェック *)
    benefit〚pair〚2〛〛 = benefit〚pair〚2〛〛 + b;
    benefit〚pair〚1〛〛 = benefit〚pair〚1〛〛 - c; If[slist〚pair〚1〛〛 <= 5,
     slist〚pair〚1〛〛 = slist〚pair〚1〛〛 + 1, slist〚pair〚1〛〛 = 5],
(* ここまで協力時の行動*)
    If[slist〚pair〚1〛〛 >= -5, slist〚pair〚1〛〛 = slist〚pair〚1〛〛 - 1,
     slist〚pair〚1〛〛 = -5]
    (* image score のレンジを[-5,5]に制限する事で無条件の協力・裏切りを定義する*)];
   benefit = benefit + c(* benefit の非負化.RandomChoice の重み用*), {m}];(*m 個のマ
ッチング反復終わり*)
```

```
 (*戦略別に利得を合計して適合度 fit に代入*)
 Table[Table[
   If[klist〚i〛 == (j - 6), fit〚j〛 = fit〚j〛 + benefit〚i〛], {j,
     1, 12}], {i, 1, n}];
 (*  適合度に基づくルーレットセレクション  *)
 klist = Table[RandomChoice[fit -> {-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6}],
{i, 1,  n}], {gene}]; (*世代 gene の反復 Do おわり*)
Print[Table[Count[klist, i - 6], {i, 1, 12}]];(* 最終的な戦略 k の分布 *)
ListPlot[Table[{i - 6, Count[klist, i - 6]}, {i, 1, 12}],
 PlotRange -> {0, 110},  Filling -> Axis, PlotStyle -> PointSize[Large]]
(* 戦略分布の可視化 *)]
```

Q.　Print を使って，途中の計算経過を確かめよ.

解答例.

```
Print[ListPlot[Table[{i - 6, Count[klist, i - 6]}, {i, 1, 12}],
  PlotRange -> {0, 110},  Filling -> Axis, PlotStyle -> PointSize[Large]]]
```

戦略分布 klist の ListPlot を Print で出力させるコマンドを{gene}の前に追加する．すると，世代毎の戦略分布の更新結果が表示される.

15．　マッチングアルゴリズム

男女のように二つの異なるグループがあり，その間でペアを作る問題はマッチング問題と呼ばれる．人と人の組み合わせである必要はなく，人と組織，組織と組織でもよい．例えば文学部の学生を希望の専修に効率よく配分する問題はマッチング問題の一種といえるし，就職活動も学生と企業のマッチング問題である．そのほかに入試，結婚など人生の様々な場面でマッチング問題にわれわれは遭遇している．このとき，割り当てられたペアから逸脱しようとする男女の組がないとき，**安定なマッチング**と呼ばれる．安定マッチングは提携ゲーム（coalitional game）のコアの一つの応用と考えられる（利得が譲渡できないのでNTU ゲームのコアである）．安定なマッチングは常に存在し，ゲールとシャプレーによって考案されたアルゴリズムによって達成できることが知られている．→ppt 参照

マッチング現象を初めて数学の問題として定式化し，誰もが一番ふさわしい相手とマッチできる「安定配分の理論」を生み出したのがシャプレーと故デヴィッド・ゲールである．わずか 7 ページの論文は，のちに GS アルゴリズムとして世に広く知られるようになった．

D. Gale, and L. S. Shapley. 1962, “College Admissions and the Stability of Marriage,” *The American Mathematical Monthly*, Vol. 69, No. 1. Jan: 9-15.

2012 年のノーベル経済学賞は「（マッチング問題における）安定配分の理論とマーケットデザインの実践に関する功績」を讃えて，アルビン・ロス（ハーバード大）とロイド・シャプレー名誉教授（カルフォルニア大学）に授与された．

ちなみに東北大生は JSTOR で，この論文を含む多くの歴史的名論文をダウンロードできるので活用しないと損だ．

問題．次の選好を持つ男女を GS アルゴリズムによってマッチングしなさい．男性側からプロポーズする場合と，女性側からプロポーズする場合とで結果がどう異なるのかを比較しなさい

男性 1 の選好　女 2＞女 3＞女 1	女性 1 の選好　男 3＞男 1＞男 2
男性 2 の選好　女 2＞女 1＞女 3	女性 2 の選好　男 3＞男 1＞男 2
男性 3 の選好　女 2＞女 1＞女 3	女性 3 の選好　男 2＞男 3＞男 1

問題．マッチングの結果が正しいことをプログラムによって確認しなさい．

手順.

- 関数 f2 と関数 gs3 を読み込む.
- 引数 proposer にプロポーズする側の選好をテンソルの形で入力する．引数 catcher にプロポーズされる側の選好を入力する.
  - 例えば 3 対 3 マッチングの場合，プロポーズする側の選好を{{2,3,1},{1,3,2},{3,1,2}}という形で入力する
- 引数 n は組数を入力する．3 対 3 マッチングの場合，n=3 となる

以下に GS アルゴリズムの *Mathematica* コードを示す.

```
f2[x_] := Module[{n},
   n = Length[x];
   Table[Flatten[
     Table[Position[x[[i]], j], {j, 1, n}]], {i, 1, n}]
  ];

gs3[n_, proposer_, catcher_] := Module[{
   girl(*男性の好み*), rank(*女性の好み *),
   position(* while ループのカウンタ用*), boy(* 仮マッチした男性のリスト*),
   gid(*男性が結婚を申し込む女性のid*), s(* 結婚を申し込む男性id *),
   b, t},
  girl = proposer; (*  男性の好みは引数proposerをそのまま入力  *)
  rank = f2[catcher]; (* 女性の好みは引数catcherをf2により変換 *)
  Print["男性の好み ", girl]; Print["女性の好み(並び替え後) ", rank];

  Table[rank[[i]] = Append[rank[[i]], n + 1], {i, 1, n}];
  (* 一番悪い順位を初期値用に追加．Cのソースでは配列が0番から始まる都合上，0列目に入る*)
  position = Table[0, {n + 1}];
  boy = Table[n + 1, {n}]; (* while条件用　番人sentinel *)
  (*Print["仮マッチリストの初期状態 ",boy];*)

  For[b = 1, b ≤ n, b++,
   s = b;
   While[s ≠ n + 1,
    gid = girl[[s]][[++(position[[s]])]]; (*  男性が，次の好みの女性に申込むためのカウンタ．
     prefix incrementだから，増加後の値1からカウンタをスタート．
     while条件を満たす，つまり仮マッチのリストに入るまで，カウンタが回る  *)
    If[rank[[gid]][[s]] < rank[[gid]][[boy[[gid]]]],
     t = boy[[gid]]; (*  一回目だけt=n+1，入れ替わるとt=前の男ID  *)
     boy[[gid]] = s; Print["仮マッチした男性のid", boy]; (* 仮マッチした男性IDを更新  *)
     s = t(* s=ふられた男IDに更新 *)     ]
   ](* while *)
  ]; (*for*)
  Table[{boy[[i]], i}, {i, 1, n}]
 ](* 出力は{求婚側，　求婚される側} の順番になっている *)
```

GS アルゴリズムは安定なマッチングを導くが，男性側がプロポーズする場合と，女性側がプロポーズする場合とで，結果が一般には異なる．

なお上記のコードは奥村晴彦氏の『C 言語による最新アルゴリズム事典』より，C 言語で書かれたソースを *Mathematica* コードに翻訳したものである．この本には他にも芸術的なコードが多数掲載されているので，プログラミングのよい勉強になる．

15.1. 応用：POP アルゴリズム

GS アルゴリズムを応用して復興公営住宅マッチングについて考える．
→スライド参照

16. トイモデルをつくろう

シミュレーションやプログラミングを理解する上で，もっとも有効な方法は，自分が計算したいこと，を試しにやってみることである．《こういう現象をモデルで表現したい》とか《こういう条件の下で計算した結果を知りたい》等の動機を持つことが，上達への近道である．

ただし数理モデルを作る上で，以下の点に気をつける必要がある．

1. 結果が自明ならば計算する必要はない．また日常言語で表現できることを，わざわざ数学で言い換える必要はない
2. 数理モデルは探索的分析には向いていない．可能な限りシンプルな仮定から，非自明な結論を導く事が数理モデルの目的である
3. シミュレーションは解析的モデルへといたるプロセスと考えるべきである
4. 数理モデルにおいては，変数の数は少なければ少ないほどよい．統制変数が多ければ多いほど欠落変数バイアスが除去できる統計モデルとは発想が逆である点に注意すること

Q　自分で説明したい現象を探し，モデルをつくってみよう