計算機支援数学：メンデルの法則
-偽造疑惑は本当か

松本　眞 m-mat@math.sci.hiroshima-u.ac.jp

平成 20 年 9 月 30 日

1　メンデルの法則

というのはご存知でしょう。エンドウ豆を例にとる。

丸い豆だけができるエンドウを何代にも渡って純化する。一方、しわしわの豆のエンドウを同様に純化する。

純粋な形質をもつ二つの親（丸、しわ）をかけ合わせる。すると、できた種を撒いて得られた個体（雑種第一代 F_1, 子のラテン語は filius）は全て丸い種子を実らせる（優性の法則)。次に、それら同士をかけ合わせた種子から得られた個体（雑種第二代 F_2）に実る種子は、丸いものばかりがなる個体としわのものばかりがなる個体がまざり、その比はおよそ 3:1 である（分離の法則)。

メンデルは、これを

$$AA \times aa = 4Aa; \quad Aa \times Aa = AA + 2Aa + aa$$

という式で説明し、なにか「遺伝子」がペアとなっているのだと結論した。

2　疑惑

メンデルは修道院で日々エンドウをまき、膨大なデータを集めて上の結論を導いた。次ページは、メンデルの実験結果である。（僕が中学生のころの教科書、「図説生物 I」実教出版からのコピー。）これに対し、統計学者 Fischer[1] は 1936 年、「この実験結果はあまりにも 3：1 に近すぎる」ということを数学的に証明した。（次の次のページ。参考:[2] に記述がある。）

3 遺伝
親から子へ

昔から，“ウリのつるにはナスはならない”といわれる。このことわざは，子は親に似ることをさしている。しかし反対に，“トンビがタカをうんだようだ”ともいい，子が親に似ない場合もあることをあらわすことわざもある。

いっぱんに，親の素質(形質)が子に伝わる現象を**遺伝**といい，同じ種類の生物の個体間で形質がちがっていることを**変異**とよんでいる。

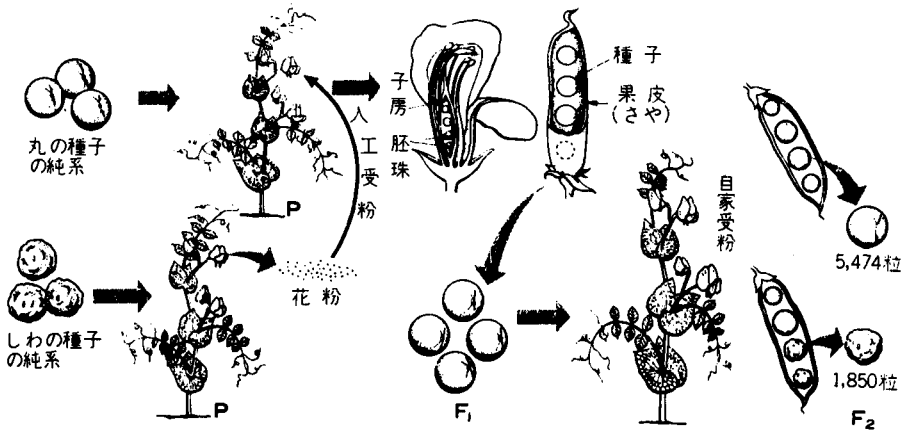
このような遺伝や変異には，どのようなきまりがあるのだろうか。

1. 遺伝の法則

かけあわせ(**交雑**)によって，家畜や栽培植物の品種を改良しようとする試みは，古くから行われていた。このことに興味をもっていたメンデルは，一つずつの形質に注目し，交雑によって得た雑種の形質が子孫にどのように伝わっていくかを調べ，はじめて，遺伝のしかたに法則性があることをみいだした。

単性雑種の遺伝(1遺伝子雑種の遺伝)

メンデルは，自家受粉をくり返し，いつも同じ形質の子孫が得られる系統(**純系**)のエンドウを選び出した。そして，一つだけ，互いにはっきり区別できる形質(**対立形質**)をもった純系どうしを親(P)[1]として，交雑して得られる**雑種第1代**(F_1)[2]や，F_1の自家受粉によって得られる雑種第2代(F_2)における形質のあらわれかたを調べ，表2のような結果を得た。メンデルは，これによってF_1にはいつでも一方の親の形質だけがあらわれることを発見した。



エンドウは，①栽培がやさしく，1世代が比較的短い。②多くの系統があり，その雑種がよく実を結ぶ。③自家受粉を行う植物で，コン虫などによる自然交雑がおこりにくいなど，遺伝の実験材料としてすぐれている。

図30　エンドウを使っての交雑実験

表2　メンデルの実験結果　(単性雑種の場合)

形質	Pの組み合わせ	F_1の形質	F_2の形質		
			優性	劣性	分離比
種子の形	丸×しわ	丸	5,474	1,850	2.96：1
子葉の色	黄色×緑色	黄色	6,022	2,001	3.01：1
種皮の色	灰色×白色	灰色	705	224	3.15：1
さやの形	ふくれ×くびれ	ふくれ	882	299	2.95：1
さやの色	緑色×黄色	緑色	428	152	2.82：1
花のつきかた	側生×頂生	側生	651	207	3.14：1
茎の高さ	高性×低性	高性	787	277	2.84：1

● メンデルの法則の発見

メンデルが遺伝の法則の発見に成功したのは，エンドウという対立形質のはっきりした，自家受粉を行う材料を選び，多くの形質のなかから一つずつの形質に注目してその遺伝を調べ，実験結果を判断するのに多数の記録をとって，統計的に処理したことなどをあげることができる。

メンデルは，実験結果を「植物雑種に関する研究」という論文にまとめ，1865年に発表した。しかし，当時の学界には認められず，1900年になって，ド・フリース，コレンス，チェルマックによってそれぞれ別々に再発見されるまで，その業績はうずもれてしまっていた。

1)，2)　親の代はラテン語の parens を略してP，子の代は filius を略してFという記号であらわし，雑種第1代はF_1，第2代はF_2というように示す。

期待値からのずれを示す自由度5のカイ2乗値は，実際のデータの場合，適度の値であった．架空データ(A)の場合，学生達はランダムな場合に比べて男女のバランスをとるようにしたためカイ2乗値は大きく，2と3の値が偶然に起こり得る値より大きくなった．架空データ(B)の場合，カイ2乗値は信じられないほどに小さく，学生たちがデータを既知の期待値に合わせようとしていたことが読みとれる．

つぎに，遺伝学の基礎として今日広く知られている遺伝形質の法則を生んだメンデルによる実験の原データについて考察しよう．

フィッシャー(Fisher[9], pp. 115-137)は，その注目すべき研究の1つにおいて，いくつかの実験におけるメンデルの理論と測定データとのずれについてのカイ2乗統計量を求めた．その結果を表3.4に示す．表から明らかなように，それぞれの場合における上側確率はきわめて高く，理論に適合するようにデータが捏造されていたという強い疑いがもたれた．

さらに，5つの実験全体をまとめた場合の適合度の確率は

$$1-0.99993=7/100,000$$

ときわめて小さな値になる．フィッシャーはこの結果についてつぎのように論評を加えている．

「満足すべき説明とはいえないが，どのような結果が得られることが

表 3.4　メンデルの実験における期待値からのズレを示す χ^2 値と確率

仮説検定のための実験	自由度	χ_0^2 (観測値)	$P(\chi^2>\chi_0^2)$
3：1の比	7	2.1389	0.95
2：1の比	8	5.1733	0.74
2因子	8	2.8110	0.94
遺伝比	15	3.6730	0.9987
3因子	26	15.3224	0.95
計	64	29.1186	0.99987
植物による変動	20	12.4870	0.90
計	84	41.6056	0.99993

(出典：R. A. Fisher[9])

望ましいかを熟知していた数名の研究補佐員によって，メンデルは欺かれていた可能性がある．この可能性は，実験データのほとんど(すべてではない)がメンデルの期待にきわめてよく合致するように偽造されていたという別の証拠によっても支持されるのである」

ホールデン(Haldane[14])は遺伝学者によって報告された，仮定された理論にきわめてよく合致するデータを例示している．ホールデンは，実験者がデータの捏造を発見するために統計家がどのような検定を用いるかについて熟知しているとすれば，実験者はそれらの検定によってデータが疑わしいと見破られないようにデータの捏造を行い，標本誤差の範囲で想定されている理論がデータに適合することを主張するであろうと指摘し，これを2次の捏造(second order faking)と命名した．いま，ある2つの事象が3：1の割合で出現するという理論があったとしよう．このとき，2つの事象の出現の割合として3：1から著しく離れた値も近い値も選ばず，そのため，理論からのずれを示すカイ2乗値は大きすぎも小さすぎもしない値をとるようにすることは，常に可能である．しかし，このような2次の捏造を検出する統計的検定が存在する．

私はかつて同僚の一人である科学者にTとHの出現頻度が1：1になるという理論を支持し，しかも捏造の疑惑を生じさせないような架空の50個のTとHの系列を記述するよう依頼したところ，彼はつぎのような系列(29個のHと21個のTからなる)を提示した．

T H T H T H H T H H
H T T H T H T H H H
T H H H T H T H T T
H H T T H T T H H H
T H H T T H H H T H

TとHの出現頻度が1：1からどの程度ずれているかを示すカイ2乗統計量の値は

$$\chi^2=(29-25)^2/25+(21-25)^2/25=1.28$$

僕の担当する授業では、かならずしも Fischer の用いた方法にこだわらずに「この実験結果はあまりにも３：１に近すぎる」かどうかを議論して行きたいと思う。

Fischer の用いたデータはメンデルの元論文から集められたもので、別紙 1 はそのごく一部に過ぎない。（Fischer の用いたデータが入手できそうだと言う人は、メールでご連絡ください。）

一方、次はとあるホームページから無断で借用してきた、とうもろこしから得られたメンデルの法則を示すデータである。

	実際の数	予想の数	実際の割合	予想の割合
黄色・丸	３４８	３５０	７．７	９
黄色・凹	１１６	１１７	２．５	３
紫色・丸	１１４	１１７	２．５	３
紫色・凹	４５	３９	１	１

一方、次はホームページ

http://www5a.biglobe.ne.jp/~kaiko/mago-13.html

から無断で借用してきた、カイコから得られたメンデルの法則を示すデータである。

	黒い体	グレーの体	白い体	合計
こぶなし	2	5	2	9
こぶあり	8	17	6	31
合　計	10	22	8	40

★まご　P　（５５頭）こぶこと KP 過剰肢の孫

こぶ有り　：　こぶ無し
42　頭　：　13　頭　　だいたい３：１

腹肢 5 対　：　腹肢 4 対
37　頭　：　18　頭　　★２：１ですね。

これらのデータは、妥当なものなのだろうか？

3　妥当性・偽造性を示す方法

3.1　シミュレーション

コンピュータプログラムを書いて、乱数を用いてこれらの実験のシミュレーションを行って見る。果たして、実験結果は 3:1 や 9:3:3:1, (1:2:1):(3:6:3) という比率にどれくらい近いだろうか。

3.2　確率計算

力づくの確率計算で、「誤差がこれ未満」という計算を行って見る。シミュレーション結果と比べてみる。

「誤差」をどう定義するか?差の自乗和? 差の絶対値の和?

3.3　カイ自乗検定

メンデルのデータのような、大きなデータを力づくで確率計算するのは困難である。

カイ自乗検定を用いて、メンデルのデータを解析してはどうか？（これは、カイ自乗検定についてすでに知識のある学生を対象とした問いである。データは別紙 2 参照。）

4　研究課題

果たして、メンデルの実験結果は偽造なのか。

メンデルの法則にまつわる実験結果について、計算機を用いて何らかの方法で検証・検討を行い、報告せよ。報告は口頭で 2008 年 10 月 30 日（木）の授業にて行う（各人発表時間 10 分程度）ほか、A4 のレポート用紙にまとめて 11 月 6 日（木）16 時までに数学事務室まで提出、という予定にしている。

具体的には、例えば次のような項目から一つ選べば十分である。

- メンデルの実験などの実験結果 R をどこかからか入手し、その実験を計算機で追試してみる。
- 上記の追試を何回か行い、R よりも誤差が小さくなる回数を数えてみる。
- 上のような実験を行うプログラムの開発，改良。

- シミュレーションではなく、R よりも誤差が小さくなる確率を明示的に求めてみる（C や Mathematica により、確率を求めるプログラムを書く。）
- 統計で「χ^2-検定」を習った人は、χ^2-検定を用いて見るのも良い。計算機実験などよりずっと簡単で効率が良い（ことが多い）。

  が、サンプルサイズが小さいとき（カイコの実験例など）では有効でないことがある。
- もっと根源的な考察。たとえば、種子のレベルでは正確に 1:3 になるモデルもありうる。我々のシミュレーションのモデルが間違っている可能性を考え、メンデルの実験結果が実は偽造ではなくなるようなモデルを構築し、計算機 and/or 数学を用いてそのモデルにおけるメンデルの結果の妥当性を示せ。

5 授業の進め方

受講生の間で、計算機に対する習熟度に大きな差があると考えられる。授業は「プログラムを自分で組んだことのない学生」のレベルに合わせて行われるので、プログラム経験のある学生にとっては意味がほとんどない。

そこで、プログラム経験のある学生は、課題の内容を理解したらあとは勝手にプログラムをしたりネットを調べたりして研究課題を遂行して欲しい。例えば僕は、「卵細胞が減数分裂後に全部生き残る」というモデルを立てた場合、メンデルの実験結果が偽造であるという話が覆されるのではないかと期待している。

もし、えんどう豆で生物学的な追試を行いこのような仮説が検証されたら、「メンデル偽造疑惑は濡れ衣であった」というインパクトのある研究となりうる。

一方、プログラミング経験のない学生向けに、ダウンロード可能な C のコードをいくつか用意した。ので、いじって慣れて欲しい。

6 メンデルの実験をシミュレートするプログラム

6.1 homepage

以下のページからプログラムがたぐりよせられます。

`http://www.math.sci.hiroshima-u.ac.jp/~m-mat/TEACH/CAM/cam.html`

6.2 Cのコード

mendel.c のコード

```
#include <stdio.h>
int main(void)
{
    int x,i;
    int occur[2]={0,0};
    printf("input seed>");
    scanf("%d",&x);
    srand(x);
    for (i=0; i<10000; i++) {
      if (rand() % 4 == 0) {occur[0]++;}
      else {occur[1]++;}
    }
    printf("(inferior,superior)=%8d, %8d\n", occur[0], occur[1]) ;
}
```

6.3 実行のしかた

1. エディタを開く
2. ホームページからプログラムをコピー・ペーストして、mendel.c というファイルとして保存する
3. シェルから gcc mendel.c を実行する
4. ./a.out を実行する
5. 初期値を入力する
6. シミュレーションの結果が出る

6.4 実行例

```
[matumoto@localhost MENDEL]$ ls
mendel.c
[matumoto@localhost MENDEL]$ gcc mendel.c
[matumoto@localhost MENDEL]$ ls
```

```
a.out   mendel.c
[matumoto@localhost MENDEL]$ ./a.out
input seed>3
(inferior,superior)=    2540,     7460
[matumoto@localhost MENDEL]$ ./a.out
input seed>5
(inferior,superior)=    2456,     7544
[matumoto@localhost MENDEL]$
```

6.5 いじくってみる

1. 10000 を 1000 とかにすると何が起きるか?
2. 1:3:3:9 で実験するにはどうすればいいか? (mendel2.c)
3. 実験を自動的に 10 回繰り返すにはどうすればいいか? (mendel3.c)
4. どうすれば、メンデルの実験ができすぎかどうか判定できるか?

mendel2.c のコード

```
#include <stdio.h>
int main(void)
{
    int x,i;
    int occur[4]={0,0,0,0};
    printf("input seed>");
    scanf("%d",&x);
    srand(x);
    for (i=0; i<10000; i++) {
      int y;
      y = rand();
      if (y % 16 == 0)     {occur[0]++;}
      else if (y % 16 < 4) {occur[1]++;}
      else if (y % 16 < 7) {occur[2]++;}
      else {occur[3]++;}
    }
    printf("%8d:%8d:%8d:%8d\n",
   occur[0], occur[1], occur[2], occur[3]) ;
}
```

mendel3.c のコード

```
#include <stdio.h>
int main(void)
{
    int x,i,j;
    int occur[2]={0,0};
    printf("input seed>");
    scanf("%d",&x);
    srand(x);
    for (j=0; j<10; j++) {
      occur[0]=0;
      occur[1]=0;
      for (i=0; i<10000; i++) {
        if (rand() % 4 == 0) {occur[0]++;}
        else {occur[1]++;}
      }
      printf("(inferior,superior)=%8d, %8d\n", occur[0], occur[1]) ;
    }
}
```

7　メンデルの実験ができすぎか?のシミュレーション

以下のプログラムも、上述のページからたぐりよせられる。

7.1　C のコード

testmendel.c のコード

```
#include <stdio.h>

double gosa(a,b)
int a, b;
{
  return (fabs(a-((double) a+b)/4));
}
```

```
int main(void)
{
    int x,y,z,i,j;
    int occur[2];
    int lessn, moren;

    printf("input seed>");
    scanf("%d",&x);
    srand(x);

    printf("input (inferior,superior)\n");
    printf("number of inf>");
    scanf("%d", &y);
    printf("number of sup>");
    scanf("%d", &z);

    lessn = 0;
    moren = 0;
    for (j=0; j<1000; j++) {
      occur[0]=0;
      occur[1]=0;
      for (i=0; i<y+z; i++) {
        if (rand() % 4 == 0) {occur[0]++;}
        else {occur[1]++;}
      }
      if (gosa(y,z) <= gosa(occur[0],occur[1])) {
        moren ++;
      } else {
        lessn ++;
      };
    }
    printf("(lessn,moren)=%8d, %8d\n", lessn, moren) ;
}
```

7.2 実行のしかた

1. ネットスケープなどを開く

2. ホームページからプログラムを探し、「セーブ」して testmendel.c というファイルとして保存する
3. マウスボタン右クリックで kterm を立ち上げる
4. kterm の中から gcc testmendel.c を実行する
5. ./a.out を実行する
6. 初期値を入力する
7. 劣性遺伝の観察個数、優性遺伝の観察個数を入力する
8. 1000 回シミュレーションを繰り返し、7 で入力した結果より誤差が小さいシミュレーション結果の回数を lessn, 大きい回数を moren として出力する。

7.3 実行例

```
[matumoto@localhost MENDEL]$ gcc testmendel.c
testmendel.c: In function ‘gosa’:
testmendel.c:6: warning: type mismatch in implicit declaration for
built-in function ‘fabs’
[matumoto@localhost MENDEL]$ ./a.out
input seed>3
input (inferior,superior)
number of inf>2001
number of sup>6022
(lessn,moren)=      94,      906
[matumoto@localhost MENDEL]$
```

これは、メンデルの実験のうち、「劣性 2001、優性 6022 本」という実験を 1000 回計算機追試してみると、メンデル以上に 1：3 に近い値を得た回数はわずか 94 回であったことを示している。

7.4 いじくってみる

1. カイコの例の「13：42」はできすぎか?
2. 「18：37」は、実験に問題があったと言えるか?

3. 「1:3:3:9」に対して上の手法を行うにはどうすれば良いか?

4. カイコの例の「1:2:1:3:6:3≈2:5:2:8:17:6」に対してはどうすれば良いか?(testkaiko.c 下のコード)

```
#include <stdio.h>
#include <stdlib.h>

#define N 6

double gosa(int a[], double b[])
{
  int i;
  double res;
  res = 0;
  for (i=0; i<N; i++) {
    res = res + fabs(((double) a[i])-b[i]);
  }
  return (res);
}

int main(void)
{
    int x,i,j,k, total_life, denomi;
    int occur[N];
    int res_to_test[N]={2, 5, 2, 8, 17, 6};
    double prob_ratio[N] = {1, 2, 1, 3, 6, 3};
    double prob_list[N];
    double prob_dist[N] = {0,0,0,0,0,0};
    double ideal[N];
    int lessn, moren;

    printf("input seed>");
    scanf("%d",&x);
    srand(x);

    total_life = 0;
    for (i=0; i<N; i++) total_life += res_to_test[i];
```

```
denomi = 0;
for (i=0; i<N; i++) denomi += prob_ratio[i];

prob_list[0] = prob_ratio[0]/denomi;
ideal[0] = prob_list[0] * total_life;
for (i=1; i<N; i++) {
  prob_list[i] = prob_list[i-1] + (prob_ratio[i]/denomi);
  ideal[i] = (prob_ratio[i]/denomi) *  total_life;
}

/*
for (i=0; i<N; i++) {
  printf("prob_list[%d]=%6f\n",i,prob_list[i]);
}
*/

lessn = 0;
moren = 0;
for (j=0; j<1000; j++) {
  for (k=0; k< N; k++) occur[k]=0;

  for (i=0; i<total_life; i++) {
      double ran;
      ran = ((double) rand())/RAND_MAX;
      for (k=0; k<N; k++) {
        if (ran < prob_list[k]) {
          occur[k]++;
          break;
        }
      }
  }

  /*
  printf("occur=");
  for (k=0; k<N ; k++) {
    printf("=%3d, ", occur[k]);
  }
  printf("\n");
```

```
      */

      if (gosa(res_to_test, ideal) <= gosa(occur, ideal)) {
        moren ++;
      } else {
        lessn ++;
      }
    }
    printf("(lessn,moren)=%8d, %8d\n", lessn, moren) ;
}
```

7.5 testkaiko.c の実行例

```
[matumoto@localhost MENDEL]$ gcc testkaiko.c
testkaiko.c: In function ‘gosa’:
testkaiko.c:12: warning: type mismatch in implicit declaration
for built-in function ‘fabs’
[matumoto@localhost MENDEL]$ ./a.out
input seed>3
(lessn,moren)=      20,      980
[matumoto@localhost MENDEL]$
```

これは、「2:5:2:8:17:6」なる比よりも理論値に近いような実験結果を得た回数が、1000 回の追試のうち 20 回だったことを意味している。

参考文献

[1] Fisher, R. A. (1936) Has Mendel's work been rediscovered? Annals of Science 1, 115-137.

[2] Rao, C. R. 邦訳： 統計学とは何か・偶然を生かす, 丸善、1993、藤越康祝他訳