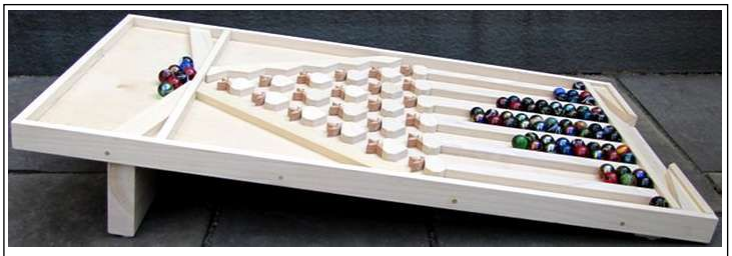
11：個別の“偶然”は予測不能ですが‥

2016 年 04 月　加藤 厚

11-1 その“全体像”は予測できます。

右のような分岐の確率が五分五分の盤（Galton Board / Pascal's marble run）に多数の玉を落とすと（個々の玉がどこに落ちるかは予測できませんが）結果の“全体像”はほぼ常に中央が高く左右に裾を引く“ベル型”※になります。その理由は、「右に落ちる確率と左に落ちる確率が等しい」なら、右ばかりあるいは左ばかりが連続することは「めったに無い」からです。※p=0.5 の二項分布。玉数が多ければ正規分布に近似。



資料 27　“偶然”の全体像の予測可能性を実証する器具の一例

他方、確率が五分五分から偏る（例:25%と 75%、5%と 95%）につれて、結果としての分布の形も異なっていきます。

今回は、推測統計の基礎であるこの二項分布※をシミュレートする script を作成します。その具体的内容は「指定確率での複数回の連続試行とその結果の図示」です。※「二項」の例：白か黒か、当りか外れか、表か裏か‥。

11-2 新しい知識は「全く不要」。

なぜなら、今回の script は既出の命令などの組み合わせのみで実現可能だからです。

form における出力用 textarea、実行用 button、入力用 text などはこれまで何度も使ってきました。

script では、まず総個数（個）、結果の位置（位[i]）、そして出力用の目盛（目）や数値軸（軸）などの値・内容を指定します。

```
<title>ゴールトン盤</title><!— saved from url=(0008)about:internet —>
<form>
   <textarea rows="16" cols="64"></textarea><br>
   投入：<input type="button" value="一個ずつ" onClick="次()">
      【<input type="button" value="まとめて（追加）" onClick="括()">
   <input type="text" size="2" value="20">個】
     下への確率<input type="text" size="2" value="50">%
</form>
<script>
   個=0;位=new Array();for(i=5;-6<i;i—)位[i]=0
   目="  0    5   10   15   20   25   30   35   50   45   50"
   軸="  +————+————+————+————+————+————+————+————+————+————+"
function 次(){
   個++;値=0;率=document.forms[0].elements[4].value
   for(i=1;i<11;i++){値++;if(Math.random()<率/100)値-=2}
   図="";位[値/2]++;図=図+目+"¥n"+軸+"¥n"
   for(i=5;-6<i;i—){星="";for(j=0;j<位[i];j++)星=星+"*"
      空="";if(-1<i)空=" ";図=図+空+i+":"+星+"  ("+位[i]+")¥n"
   }
   document.forms[0].elements[0].value=図+軸+"¥n"+目+"¥n 計："+個+"個"
}
function 括(){
   限=document.forms[0].elements[3].value;for(k=1;k<=限;k++){次()}
}
</script>
```

位[i]の i=5〜-5 に初期値=0 を代入する。

Math.random の乱数は 0 以上 1 未満。
値-=2 は値=値-2 と同義。

資料 28　ゴールトン盤（＝二項分布）をシミュレートする JavaScript の一例

11-3 一個の玉を扱う function 次()

「一個ずつ」のボタンをクリックする毎に function 次()が 1 回働きます。まず総個数を一増して値に 0 を代入し、続いて「値を一増の上、発生させた乱数が『下への確率』の値以下なら値を二減」※という処理を 10 回行います＃。

※結果は+1 か-1。＃結果は-10, -8, -6, -4, -2, 0, 2, 4, 6, 8, 10 のいずれか。

そして、位[10 回終了時の値/2]を一増し、一増後の数だけ *を連結の上、軸・目・空などで書式を整えた内容を図に代入（＝連結）していき、総個数とともに出力（textarea に代入）します。

11-4 多数の玉を一括処理する function 括()

「まとめて（追加）」ボタンをクリックする毎に function 括()が 1 回働きます。その内容は「要素[3]で指定された回数＝限‥初期値=20 だけ次()を実行する」です。

p が小さい場合、玉数を増やすと結果は「ポアソン分布」に近づきます。試してみましょう。

資料 29　資料 28 の内容の実行結果の一例

【発展課題】

玉数が増えても横幅が溢れないようにするには？

㉑function では別 function の呼び出しも可能&便利。
㉒重複する for では異なる回数記号(i, j‥)を使用。