LISP-STATによる無作為標本抽出のコンピュータ・シミュレーション: 無限母集団の場合 (v00-16)

上條哲男 (上智大学経済学部)

2004 年 (平成 16 年)7 月 12 日

1 はじめに

無限母集団からの無作為標本抽出をシミュレーションで行う方法を説明する。

単純無作為標本抽出法および有限母集団からの無作為標本抽出をシミュレーションで行う方法については、上條 (2004) を参照。

使用するコンピュータ・プログラミング言語は LISP である (栗原 (1993), Steele(1990), 竹内 (1986), Winston and Horn(1989), および湯浅，萩谷 (1986) などを参照)。プログラムの作成および実行には、LISP-STAT を利用している (竹村 (1997), 垂水 (1999), および Tierney(1990) などを参照)。ミネソタ大学の School of Statistics の Luke Tierney が LISP-STAT を考案し、コンピュータ・ソフトウエアである XLISP-STAT を開発した。XLISP-STAT は、LISP 言語の方言の 1 つである XLISP 言語を使用している。

シミュレーションについては、Bratley, Fox, and Schrage(1987), Freund and Williams(1977, 11; 邦訳 (1974, 8)), 藤曲哲郎，森 隆一，山本 浩 (1996), Hammersley and Handscomb(1964), 石川 宏 (1994), 木下宗七編 (1996, 第 7 章), Law and Kelton(1991), 宮武 修，脇本和昌 (1978), Naylor, Balintfy Burdick and, Chu(1966), Pidd(1998), Rubinstein and Melamed(1998), および津田孝夫 (1995) などを参照。

つぎの 2 においては、サイコロ投げのシミュレーションについて、3 においては、コイン投げのシミュレーションについて説明する。それぞれにおいて、第

1 に、実験結果に数値を対応させ、乱数表を見ながら行うハンド・シミュレーション、第 2 に、分布に基づいて、乱数表を見ながら行うハンド・シミュレーション、そして第 3 に、分布に基づいて、LISP-STAT で行うコンピュータ・シミュレーションについて説明する。

2　サイコロ投げのシミュレーション

均一なサイコロ (6 面体のサイコロとする) を 30 回 (実験回数、抽出・標本の大きさ) 投げて、 1 の目、 2 の目、 ... 、そして 6 の目がそれぞれ何回出るか、というシミュレーションを行う方法を説明する。各回で各目が出るのは同様に確からしい、と仮定する。

つぎの 2.1 においては、実験結果に数値を対応させ、乱数表を見ながら行うハンド・シミュレーションについて、2.2 においては、分布に基づいて、乱数表を見ながら行うハンド・シミュレーションについて、そして、2.3 においは、分布に基づいて LISP-STAT で行うコンピュータ・シミュレーションについて説明する。

2.1　実験結果に数値を対応させ、乱数表を見ながら行うハンド・シミュレーション

サイコロの 1 の目に数値 1 を、 2 の目に数値 2 を、... 、そして 6 の目に数値 6 を対応させる。1 桁の乱数を利用して、 1 が抽出された場合には 1 の目が、 2 が抽出さた場合には 2 の目が、... 、そして 6 が抽出された場合には 6 の目が出たことにする。この場合、サイコロの目は、 1 から 6 までであるから、0 および 7 から 9 までの 4 個の乱数は使用せずに捨てることになる。すなわち、捨てる割合は $4/10 = 0.4$ であり、平均して、 40%の乱数を捨てることになる。なお、乱数が 0 および 7 から 9 までの場合には、目の数の欄に-(hyphen) を記入しなさい。

付録 A の「9 桁の乱数表」を使用することにすると、第 1 行、第 1 列の乱数 218418296 の第 1 桁の数字 2 を出発値として、つぎに、その行の第 2 列の乱数 956317576 の第 1 の数字 9 を抽出し、さらに、その行を横に順番に抽出し、右端の列に来た場合には、下の第 2 行に進む。このようにして、実験回数まで抽

出を繰り返す。実際には、9 桁の乱数表から第 2 桁以降を切り捨てることによって作成された「1 桁の乱数表」(付録 B を参照) を使用することにする。

抽出された乱数およびサイコロの目の数を表示すると、表 1 「サイコロ投げのハンド・シミュレーションの結果：乱数に目の数をそのまま対応させる」のようになる。

表 1 の「サイコロ投げのハンド・シミュレーションの結果：乱数に目の数をそのまま対応させる」において、

$$
\begin{aligned}
&\text{抽出した乱数の大きさ} &&= \quad 60 \\
&\text{捨てた乱数の大きさ} &&= \quad 30 \\
&\text{捨てた乱数の割合} &&= \quad \text{捨てた乱数の大きさ/抽出した乱数の大きさ} \\
& &&= \quad 30/60 \;=\; 0.500 \quad (50\%)
\end{aligned}
$$

であり、実際には 50%の乱数を捨てたことになる。

2.2 分布に基づいて、乱数表を見ながら行うハンド・シミュレーション

(1) 小数第 1 位まで計算された確率の値を使用する。

確率を計算し、小数点第 2 位以下を切り捨て、小数第 1 位の値を使用することにする（表 2 を参照)。

各回で、サイコロの各目が出る確率は、 $p = 1/6$ であるとする。まず、1/6 を計算し、小数点第 2 位以下を切り捨てると 0.1 となる。つぎに、累積確率およびその累積確率の $10(= 10^1)$ 倍を計算する。このことから、 1 桁の乱数を使用すればよいことがわかる。

「この場合におけるハンド・シミュレーション」は、 2.1 の「実験結果に数値を対応させ、乱数表を見ながら行うハンド・シミュレーション 」と同じである。したがって、平均として、 40%の乱数を捨てることになる。なお、 2.1 では、実際には、50%の乱数を捨てている。

(2) 小数第 2 位まで計算された確率の値を使用する。

確率を計算し、小数点第 3 位以下を切り捨て、小数第 2 位までの値を使用することにする（表 3 を参照)。

各回で、サイコロの各目が出る確率は、$p = 1/6$ であるとする。まず、1/6

表 1: サイコロ投げのハンド・シミュレーションの結果：乱数に目の数をそのまま対応させる

抽出番号	1	2	3	4	5	6	7	8	9	10
乱数	2	9	8	5	4	0	2	1	0	6
目の数	2	-	-	5	4	-	2	1	-	6
出現番号	1	-	-	2	3	-	4	5	-	6
抽出番号	11	12	13	14	15	16	17	18	19	20
乱数	0	4	4	7	7	0	8	3	0	0
目の数	-	4	4	-	-	-	-	3	-	-
出現番号	-	7	8	-	-	-	-	9	-	-
抽出番号	21	22	23	24	25	26	27	28	29	30
乱数	8	8	1	0	2	9	1	3	8	2
目の数	-	-	1	-	2	-	1	3	-	2
出現番号	-	-	10	-	11	-	12	13	-	14
抽出番号	31	32	33	34	35	36	37	38	39	40
乱数	6	5	8	4	9	5	1	7	3	1
目の数	6	5	-	4	-	5	1	-	3	1
出現番号	15	16	-	17	-	18	19	-	20	21
抽出番号	41	42	43	44	45	46	47	48	49	50
乱数	5	3	6	3	2	7	1	2	8	8
目の数	5	3	6	3	2	-	1	2	-	-
出現番号	22	23	24	25	26	-	27	28	-	-
抽出番号	51	52	53	54	55	56	57	58	59	60
乱数	0	7	0	7	7	3	8	7	7	4
目の数	-	-	-	-	-	3	-	-	-	4
出現番号	-	-	-	-	-	29	-	-	-	30

表 2: サイコロ投げのシミュレーション： 分布に基づいて、1 桁の乱数に目の数を対応させる

目の数	確率	確率	累積確率	累積確率*10	乱数
1	1/6	0.1	0.1	1	1
2	1/6	0.1	0.2	2	2
3	1/6	0.1	0.3	3	3
4	1/6	0.1	0.4	4	4
5	1/6	0.1	0.5	5	5
6	1/6	0.1	0.6	6	6
合計		0.6			

を計算し、小数点第 3 位以下を切り捨てると 0.16 となる。つぎに、累積確率およびその累積確率の $100(=10^2)$ 倍を計算する。このことから、 2 桁の乱数を使用すればよいことがわかる。

表 3: サイコロ投げのシミュレーション：分布に基づいて、2 桁の乱数に目の数を対応させる

目の数	確率	確率	累積確率	累積確率*100	乱数
1	1/6	0.16	0.16	16	00-15
2	1/6	0.16	0.32	32	16-31
3	1/6	0.16	0.48	48	32-47
4	1/6	0.16	0.64	64	48-63
5	1/6	0.16	0.80	80	64-79
6	1/6	0.16	0.96	96	80-95
合計		0.96			

「小数第 2 位まで計算された確率の値を使用する」場合におけるサイコロ投げのハンド・シミュレーションでは、 2 桁の乱数 00 から 95 までの 96 個の乱数を使用し、96 から 99 までの 4 個の乱数を捨てることになる。すなわち、捨てる割合は $4/100 = 0.04$ であり、平均として、4%の乱数を捨てることになる。なお、乱数が 96 から 99 までの 場合には、目の数の欄

に-(hyphen) を記入しなさい。

乱数が、00 から 15 までであれば、サイコロの目の数が、 1 であり、乱数が、16 から 31 までであれば、サイコロの目の数が、 2 であり、... 、そして乱数 80 から 95 までであれば、サイコロの目の数は、 6 であるとする。使用する乱数表は、付録 C の「2 桁の乱数表」である。乱数の抽出方法は、2.1 で説明したものと同様である。

抽出された乱数およびサイコロの目の数を表示すると、表 4 「サイコロ投げのハンド・シミュレーションの結果：2 桁の乱数に目の数を対応させる」のようになる。

表 4: サイコロ投げのハンド・シミュレーションの結果：2 桁の乱数に目の数を対応させる

抽出番号	1	2	3	4	5	6	7	8	9	10
乱数	21	95	82	56	41	06	25	10	04	63
目の数	2	6	6	4	3	1	2	1	1	4
出現番号	1	2	3	4	5	6	7	8	9	10
抽出番号	11	12	13	14	15	16	17	18	19	20
乱数	06	44	40	75	79	00	89	35	09	01
目の数	1	3	3	5	5	1	6	3	1	1
出現番号	11	12	13	14	15	16	17	18	19	20
抽出番号	21	22	23	24	25	26	27	28	29	30
乱数	85	84	12	00	26	91	11	35	82	26
目の数	6	6	1	1	2	6	1	3	6	2
出現番号	21	22	23	24	25	26	27	28	29	30

表 4 の 「サイコロ投げのハンド・シミュレーションの結果：2 桁の乱数に目の数を対応させる」において、

抽出した乱数の大きさ = 30
捨てた乱数の大きさ = 0
捨てた乱数の割合 = 捨てた乱数の大きさ/抽出した乱数の大きさ
= 0/30 = 0

であり、実際には、捨てた乱数はなく、すべての乱数を利用した。

(3) 小数第 3 位まで計算された確率の値を使用する。

確率を計算し、小数点第 4 位以下を切り捨て、小数第 3 位までの値を使用することにする（表 5 を参照)。

各回で、サイコロの各目が出る確率は、$p = 1/6$ であるとする。$1/6$ を計算し、小数点第 4 位以下を切り捨てると 0.166 となる。つぎに、累積確率およびその累積確率の $1000(= 10^3)$ 倍を計算する。このことから、 3 桁の乱数を使用すればよいことがわかる。

表 5: サイコロ投げのシミュレーション：分布に基づいて、3 桁の乱数に目の数を対応させる

目の数	確率	確率	累積確率	累積確率*1000	乱数
1	1/6	0.166	0.166	166	000-165
2	1/6	0.166	0.332	332	166-331
3	1/6	0.166	0.498	498	332-497
4	1/6	0.166	0.664	664	498-663
5	1/6	0.166	0.830	830	664-829
6	1/6	0.166	0.996	996	830-995
合計		0.996			

「小数第 3 位まで計算された確率の値を使用する」場合における、サイコロ投げのハンド・シミュレーションでは、3 桁の乱数 000 から 995 までの 996 個の乱数を使用し、996 から 999 までの 4 個の乱数を捨てることになる。すなわち、捨てる割合は $4/1000 = 0.004$ であり、平均として、0.4%の乱数を捨てることになる。なお、乱数が 996 から 999 までの 場合には、目の数の欄に-(hyphen) を記入しなさい。

乱数が、 000 から 165 までであれば、サイコロの目の数は、 1 であり、乱数 166 から 331 までであれば、サイコロの目の数は、 2 であり、 ...、そして乱数が、830 から 995 までであれば、サイコロの目の数は 6 であるとする。使用する乱数表は、付録 D の「3 桁の乱数表」である。乱数の抽出方法は、2.1 で説明したものと同様である。

抽出された乱数およびサイコロの目の数を表示すると、表 6 「サイコロ

投げのハンド・シミュレーションの結果：3 桁の乱数に目の数を対応させる」のようになる。

表 6: サイコロ投げのハンド・シミュレーションの結果：3 桁の乱数に目の数を対応させる

抽出番号	1	2	3	4	5	6	7	8	9	10
乱数	218	956	829	561	415	066	257	109	043	633
目の数	2	6	5	4	3	1	2	1	1	4
出現番号	1	2	3	4	5	6	7	8	9	10
抽出番号	11	12	13	14	15	16	17	18	19	20
乱数	061	449	401	754	797	001	897	350	094	013
目の数	1	3	3	5	5	1	6	3	1	1
出現番号	11	12	13	14	15	16	17	18	19	20
抽出番号	21	22	23	24	25	26	27	28	29	30
乱数	859	840	123	007	260	912	113	351	822	267
目の数	6	6	1	1	2	6	1	3	5	2
出現番号	21	22	23	24	25	26	27	28	29	30

表 6 の 「サイコロ投げのハンド・シミュレーションの結果：3 桁の乱数に目の数を対応させる」において、

$$
\begin{aligned}
&\text{抽出した乱数の大きさ} \;=\; 30\\
&\text{捨てた乱数の大きさ} \;=\; 0\\
&\text{捨てた乱数の割合} \;=\; \text{捨てた乱数の大きさ/抽出した乱数の大きさ}\\
&\qquad\qquad\qquad\quad =\; 0/30 \;=\; 0
\end{aligned}
$$

であり、実際には、捨てた乱数はなく、すべての乱数を利用した。

2.3 分布に基づいて、LISP-STAT で行う コンピュータ・シミュレーション

サイコロ投げのコンピュータ・シミュレーションを XLISP-STAT を利用して実行することにする。

XLISP-STAT を起動し、まず、フロッピード ライブ (a:とする) に入れてあるフロッピーディスク内のフォルダ book-stat 内のサブフォルダ programs 内のサブサブフォルダ sampling-infinite-pop 内のサブサブサブフォルダ saikoro-tossing 内のプログラム・ファイル

saikoro-inc-hyphen-list-random-3-keta-fun-v01-01.lsp
（v01-01 は version 01.01 を示しており、変更されている場合がある）。
table-saikoro-inc-hyphen-random-3-keta-v01-01.lsp
（v01-01 は version 01.01 を示しており、変更されている場合がある）。

を load し、つぎに、フォルダ book-stat 内のサブフォルダ programs 内のサブサブフォルダ random-number 内のサブサブサブフォルダ random-number-seed 内にあるプログラム・ファイル

random-number-seed-v01-01.lsp
（v01-01 は version 01.01 を示しており、変更されている場合がある）。

を load する。

表 5 の「サイコロ投げのシミュレーション：分布に基づいて、3 桁の乱数に目の数を対応させる」における結果を用いてコンピュータ・シミュレーションを行う。まず、(1) サイコロ投げのコンピュータ・シミュレーションの結果の表を作成する方法を、つぎに (2) サイコロ投げのコンピュータ・シミュレーションの結果のリストを求める方法を説明する。

(1) サイコロ投げのコンピュータ・シミュレーションの結果の表を作成する方法

サイコロ投げのコンピュータ・シミュレーションの結果の表を作成するためには関数 (table-saikoro-inc-hyphen-random-3-keta no-within-class name seed

saikoro-chushutsu-size) を実行する。この関数においては、ある抽出番号における 3 桁の乱数が 996 以上である場合には、該当する目の欄には、-(hyphen) が表示されることになっている。no-within-class(クラス内番号) を 123456789、name(名前) を tetsuo-kamijo、seed(種子の初期値) を 123456789 とし、saikoro-chushutsu-size(実験回数) を 30 回として、上記関数を実行するには、促進記号（>）の後ろに、

```
> (table-saikoro-inc-hyphen-random-3-keta
    123456789 'tetsuo-kamijo 123456789 30)
```

と入力する。その結果は表 7 の「 サイコロ投げのコンピュータ・シミュレーションの結果（実験回数=30 回）」に表示されている。この結果は、表 6 の「サイコロ投げのハンド・シミュレーションの結果：3 桁の乱数に目の数を対応させる」における結果と同じである。

(2) サイコロ投げのコンピュータ・シミュレーションの結果のリストを求める方法

サイコロ投げのコンピュータ・シミュレーションの結果のリストを求めるためには関数 (saikoro-list-random-3-keta-fun seed saikoro-chushutsu-size) を実行する。この関数においては、ある抽出番号における 3 桁の乱数が 996 以上である場合には、その乱数を捨て、つぎの 3 桁の乱数を抽出することにしている。seed(種子の初期値) を 123456789 とし、saikoro-chushutsu-size(実験回数) を 30 回として、上記関数を実行し、結果のリストを sample-01-30-saikoro-01-tk（ここで、tk は tetsuo kamijo のイニシャルである）とするには、促進記号（>）の後ろに、

```
> (def sample-01-30-saikoro-01-tk
       (saikoro-list-random-3-keta-fun 123456789 30)
  )
```

と入力する。その結果、sample-01-30-saikoro-01-tk は

```
> sample-01-30-saikoro-01-tk
  (2 6 5 4 3 1 2 1 1 4
   1 3 3 5 5 1 6 3 1 1
   6 6 1 1 2 6 1 3 5 2)
```

表 7: サイコロ投げのコンピュータ・シミュレーションの結果（実験回数=30 回）

```
> (table-saikoro-inc-hyphen-random-3-keta
                0 'tetsuo-kamijo 123456789 30)

  Table of Saikoro no 'Me no Kazu'(1-6)
     If random number is >= 996,
     then 'Me no Kazu' is -(hyphen).

    no-within-class:                123456789
    Name                          TETSUO-KAMIJO
    seed:                           123456789
    Saikoro Chushutsu Size                 30

-------------------------------------------------------------------
   i     i    i+1   i+2   i+3   i+4   i+5   i+6   i+7   i+8   i+9
-------------------------------------------------------------------
   1     2     6     5     4     3     1     2     1     1     4
  11     1     3     3     5     5     1     6     3     1     1
  21     6     6     1     1     2     6     1     3     5     2
-------------------------------------------------------------------
    Random Number Chushutsu Size           30
-------------------------------------------------------------------
```

となっている。この結果は、表6の「サイコロ投げのハンド・シミュレーションの結果：3桁の乱数に目の数を対応させる」および表7の「サイコロ投げのコンピュータ・シミュレーションの結果（実験回数=30回）」における結果と同じである。表6および表7に「-(hyphen)」が含まれている場合には、その「-(hyphen)」を除いた結果と同じになるはずである。

なお、捨てた乱数の大きさは、つぎのように求められる。

```
> (- random-number-chushutsu-size *saikoro-chushutsu-size*)
0
```

この場合、捨てた乱数はなかった、という結果となっている。

上記 sample-01-30-saikoro-01-tk を、フロッピードライブ a:にあるフロッピー内のフォルダ LS-tk にファイル名 sample-01-30-saikoro-01-tk.lsp として保存するには

```
> (savevar 'sample-01-30-saikoro-01-tk
           "a:/LS-tk/sample-01-30-saikoro-01-tk")
```

とする。

XLISP-STAT を終了し、再度起動した場合に、このファイル sample-01-30-saikoro-01-tk.lsp を load すると、データとして、sample-01-30-saikoro-01-tk が使用できるようになる。

3　コイン投げのシミュレーション

均一なコインを30回 (実験回数、抽出・標本の大きさ) 投げて、表および裏が何回出るか、というシミュレーションを行う方法を説明する。各回で表あるいは裏が出るのは同様に確からしい、と仮定する。

つぎの3.1においては、実験結果に数値を対応させ、乱数表を見ながら行うハンド・シミュレーションについて、3.2においては、分布に基づいて、乱数表を見ながら行うハンド・シミュレーションについて、そして、3.3においは、分布に基づいて LISP-STAT で行うコンピュータ・シミュレーションについて説明する。

3.1 実験結果に数値を対応させ、乱数表を見ながら行うハンド・シミュレーション

コイン投げの場合、シミュレーションの結果 (実験結果) は、表あるいは裏であり、それらに数値を対応させる必要がある。例えば、表には数値 1 を、そして裏には数値 0 を対応させる、という方法が考えられる。この場合、乱数の 1 が抽出された場合には、表が出たことにし、乱数の 0 が抽出された場合には、裏が出たことにする。この方法では、乱数 1 および 0 の 2 種類の乱数を使用し、2 から 9 までの 8 個の乱数を捨てることになる。捨てることになる乱数を減らす方法を考える必要がある。以下に１つの方法を示すことにする。それは、表には数値の奇数を、そして裏には数値の偶数を対応させるという方法である。この場合、乱数が奇数の場合には、表が、乱数が偶数の場合には、裏が出たことにする（表 8 を参照)。この方法では、捨てる乱数はなく、すべての乱数を利用することになる。

表 8: コイン投げのシミュレーション：直接的に、乱数に結果を対応させる

実験結果	数値	乱数
表	1 3 5 7 9	1 3 5 7 9
裏	0 2 6 4 8	0 2 4 6 8

付録 A の「9 桁の乱数表」の第 1 行、第 1 列の乱数 218418296 の第 1 桁の数字 2 を出発値として、つぎに、その行の第 2 列の乱数 956317576 の第 1 桁の数字 9 を抽出し、さらに、その行を横に順番に抽出し、右端の列に来た場合には、下の第 2 行に進む。このようにして、実験回数まで抽出を繰り返す。実際には、9 桁の乱数表から第 2 桁以降を切り捨てることによって作成された「1 桁の乱数表」(付録 B を参照) を使用する。

抽出された乱数およびコイン投げのハンド・シミュレーションの結果（実験結果）を表示すると、表 9 「コイン投げのハンド・シミュレーションの結果：乱数にコインの表・裏を対応させる」のようになる。

表 9: コイン投げのハンド・シミュレーションの結果：乱数にコインの表・裏を対応させる

抽出番号	1	2	3	4	5	6	7	8	9	10
乱数	2	9	8	5	4	0	2	1	0	6
実験結果	裏	表	裏	表	裏	裏	裏	表	裏	裏
抽出番号	11	12	13	14	15	16	17	18	19	20
乱数	0	4	4	7	7	0	8	3	0	0
実験結果	裏	裏	裏	表	表	裏	裏	表	裏	裏
抽出番号	21	22	23	24	25	26	27	28	29	30
乱数	8	8	1	0	2	9	1	3	8	2
実験結果	裏	裏	表	裏	裏	表	表	表	裏	裏

3.2 分布に基づいて、乱数表を見ながら行うハンド・シミュレーション

各回で、表の出る確率は、$p = 1/2$ であるとする。$1/2$ は 0.5 で小数以下第 1 桁で割り切れる。したがって、1 桁の乱数を利用することになる。累積確率およびその累積確率の $10(= 10^1)$ 倍を計算する。乱数が 0 から 4 までであれば、表が出たものとし、乱数が、5 から 9 までであれば、裏が出たものとする (表 10 を参照)。この場合、捨てる乱数はなく、すべての乱数を利用する。使用する乱数表は、「1 桁の乱数表」(付録 B を参照) である。乱数の抽出方法は、3.1 の「実験結果に数値を対応させ、乱数表を見ながら行うハンド・シミュレーション」で説明したものと同様である。

表 10: コイン投げのシミュレーション：分布により、1 桁の乱数を結果に対応させる

実験結果	確率 p	累積確率	累積確率*10	乱数
表	0.5	0.5	5	0-4
裏	0.5	1.0	10	5-9

抽出された乱数およびコイン投げのハンド・シミュレーションの結果（実験

結果）を表示すると、表 11 「コイン投げのハンド・シミュレーションの結果（表、裏で表示）: 1 桁の乱数に結果を対応させる」のようになる。

表 11: コイン投げのハンド・シミュレーションの結果（表、裏で表示）: 1 桁の乱数に結果を対応させる

抽出番号	1	2	3	4	5	6	7	8	9	10
乱数	2	9	8	5	4	0	2	1	0	6
実験結果	表	裏	裏	裏	表	表	表	表	表	裏
抽出番号	11	12	13	14	15	16	17	18	19	20
乱数	0	4	4	7	7	0	8	3	0	0
実験結果	表	表	表	裏	裏	表	裏	表	表	表
抽出番号	21	22	23	24	25	26	27	28	29	30
乱数	8	8	1	0	2	9	1	3	8	2
実験結果	裏	裏	表	表	表	裏	表	表	裏	表

コイン投げのハンド・シミュレーションの結果（実験結果）において、「表」の出現した回数、その平均値などを計算する際の便宜のために、表 11 「コイン投げのハンド・シミュレーションの結果（表、裏で表示）: 1 桁の乱数に結果を対応させる」における「表」を 1 で、「裏」を 0 で表示することにする。それが、表 12 「コイン投げのハンド・シミュレーションの結果（表を 1、裏を 0 で表示）: 1 桁の乱数に結果を対応させる」 である。

3.3 分布に基づいて、LISP-STAT で行うコンピュータ・シミュレーション

コイン投げのコンピュータ・シミュレーションを XLISP-STAT を利用して実行することにする。

XLISP-STAT を起動し、まず、フロッピードライブ (a:とする) に入れてあるフロッピーディスク内のフォルダ book-stat 内のサブフォルダ programs 内のサブサブフォルダ sampling-infinite-pop 内のサブサブサブフォルダ coin-tossing 内のプログラム・ファイル

表 12: コイン投げのハンド・シミュレーションの結果 (表を 1、裏を 0 で表示):1 桁の乱数に結果を対応させる

抽出番号	1	2	3	4	5	6	7	8	9	10
乱数	2	9	8	5	4	0	2	1	0	6
実験結果	1	0	0	0	1	1	1	1	1	0
抽出番号	11	12	13	14	15	16	17	18	19	20
乱数	0	4	4	7	7	0	8	3	0	0
実験結果	1	1	1	0	0	1	0	1	1	1
抽出番号	21	22	23	24	25	26	27	28	29	30
乱数	8	8	1	0	2	9	1	3	8	2
実験結果	0	0	1	1	1	0	1	1	0	1

coin-tossing-list-generation-fun-v01-01.lsp
（v01-01 は version 01.01 を示しており、変更されている場合がある）。
table-coin-tossing-v01-01.lsp
（v01-01 は version 01.01 を示しており、変更されている場合がある）。

を load し、つぎに、フォルダ book-stat 内のサブフォルダ programs 内のサブサブフォルダ random-number 内のサブサブサブフォルダ random-number-seed 内にあるプログラム・ファイル

random-number-seed-v01-01.lsp
（v01-01 は version 01.01 を示しており、変更されている場合がある）。

を load する。

表 10 の「コイン投げのシミュレーション：分布により、1 桁の乱数を結果に対応させる」における結果を用いてコンピュータ・シミュレーションを行う。まず、(1) コイン投げのコンピュータ・シミュレーションの結果の表を作成する方法を、つぎに (2) コイン投げのコンピュータ・シミュレーションの結果のリストを求める方法を説明する。

(1) コイン投げのコンピュータ・シミュレーションの結果の表を作成する方法

コイン投げのシミュレーションの結果（実験結果）の表を作成するためには関数 (table-coin-tossing no-within-class name seed coin-chushutsu-size) を実行する。no-within-class(クラス内番号) を 123456789、name(名前) を tetsuo-kamijo、seed(種子の初期値) を 123456789 とし、coin-chushutsu-size(実験回数) を 30 回として、上記関数を実行するには、促進記号（>）の後ろに、

```
> (table-coin-tossing  123456789 'tetsuo-kamijo 123456789 30)
```

と入力する。その結果は表 13 の「 コイン投げのシミュレーションの結果 (実験結果)（実験回数=30 回)」に表示されている。この結果は、表 12 の「コイン投げのハンド・シミュレーションの結果 (表を 1、裏を 0 で表示):1 桁の乱数に結果を対応させる」における結果と同じである。

(2) コイン投げのコンピュータ・シミュレーションの結果の
リストを求める方法

コイン投げのコンピュータ・シミュレーションの結果（実験結果）のリストを求めるためには関数 (coin-tossing-list-fun seed coin-chushutsu-size) を実行する。seed(種子の初期値) を 123456789 とし、coin-chushutu-size(実験回数) を 30 回として、上記関数を実行し、結果のリストを sample-01-30-coin-01-tk（ここで、tk は tetsuo-kamijo のイニシャルである）とするには、促進記号 (>) の後ろに、

```
> (def sample-01-30-coin-01-tk
       (coin-tossing-list-fun 123456789 30)
  )
```

と入力する。その結果は

```
> sample-01-30-coin-01-tk
  (1 0 0 0 1 1 1 1 1 0
   1 1 1 0 0 1 0 1 1 1
   0 0 1 1 1 0 1 1 0 1)
```

である。この結果は、表 12 の「コイン投げのハンド・シミュレーションの結果 (表を 1、裏を 0 で表示):1 桁の乱数に結果を対応させる」および表 13 の「コイン投げのコンピュータ・シミュレーションの結果（実験結果）（実験回数=30 回)」

表 13: コイン投げのコンピュータ・シミュレーションの結果（実験結果）（実験回数=30 回）

```
> (table-coin-tossing 0 'tetsuo-kamijo 123456789 30)

  Table of Coin Tossing
     (1: Omote(Head), 0: Ura(Tail)

    no-within-class:                123456789
    Name                          TETSUO-KAMIJO
    seed:                           123456789
    Coin Chushutsu Size                    30

-------------------------------------------------------------------
   i     i    i+1   i+2   i+3   i+4   i+5   i+6   i+7   i+8   i+9
-------------------------------------------------------------------
   1     1     0     0     0     1     1     1     1     1     0
  11     1     1     1     0     0     1     0     1     1     1
  21     0     0     1     1     1     0     1     1     0     1
-------------------------------------------------------------------
```

における結果と同じである。

上記 sample-01-30-coin-01-tk を、フロッピードライブ a:にあるフロッピー内のフォルダ LS-tk にファイル名 sample-01-30-coin-01-tk.lsp として保存するには

```
> (savevar 'sample-01-30-coin-01-tk
           "a:/LS-tk/sample-01-30-coin-01-tk")
```

とする。

XLISP-STAT を終了し、再度起動した場合に、このファイル sample-01-30-coin-01-tk.lsp を load すると、データとして、sample-01-30-coin-01-tk が使用できるようになる。

付録 A　9桁の乱数表

XLISP-STAT において、下記のように、促進記号（>）の後に、
(table-random-number-9 123456789 'tetsuo-kamijo 123456789)
と入力することによって出力される。

```
> (table-random-number-9 123456789 'tetsuo-kamijo 123456789)

  Table of Random Number(9 digits)
    No within class:   123456789
    Name           TETSUO-KAMIJO
    Initial seed:      123456789

    If a digit number of random number is n(<=8),
    then add (9-n) number of zero(s) before the
    random number.
    For example, random number is 66118734(8 digits),
    then the number becomes 066118734.
```

i	i	i+1	i+2	i+3	i+4
1	218418296	956317576	829509233	561695442	415307081
6	66118734	257577792	109956793	43828997	633965712
11	61727229	449538960	401306281	754673486	797286954
16	1838371	897504060	350752337	94544750	13616891
21	859096855	840847450	123103915	7512364	260302997
26	912483707	113664046	351628659	822887316	267132270
31	692066499	561662474	861215790	453793775	911977028
36	597916877	188954691	761492056	396988475	185314117
41	574365861	367026667	617204827	361528704	212929997
46	714471214	117706867	299329148	825002954	824660073
51	61861770	710780524	88283333	777994008	745303068
56	308674919	899373090	763536724	761730646	406969640
61	938749454	562088285	17820021	501103225	41909255
66	368850581	271723601	858572561	29036558	17442279
71	152383787	114318671	353907259	119307827	206652759
76	212923957	612947552	809519074	587089634	215491643
81	768056363	723296722	448018990	855176118	945017496
86	909056924	519725721	30194625	481066955	292312883
91	902639843	667841511	412278035	156948490	833281632
96	964404164	740789914	456098861	653561422	406826949
101	540538601	832280939	145755764	717128318	775650756
106	362261623	531110845	379976752	269285263	877417819
111	761284802	913675004	135794085	291195228	118210970
116	771780843	320632915	877412951	679475764	949175543
121	793361180	21366233	102278301	991406155	563259625
126	704527526	994145554	604334274	46145179	562029006
131	21515388	609134812	728788505	748406594	469633795
136	135193808	202347838	860120225	40624381	773972507
141	155940722	895727204	487127308	148681700	893344749
146	445207315	599347217	228688778	572299614	639618372
151	65984537	2119914	629405178	412841577	628397089
156	469879128	258508626	754493277	768513356	403978118
161	660235871	584298490	304735190	684344665	780800314
166	910878278	131230770	595554847	490322732	854158896
171	848579945	83142974	383971625	411112335	565026143
176	394391505	538036249	775243041	509805754	305324214
181	584075669	559778367	195029220	856103379	529502248
186	344292860	530100924	406241019	692813821	121905097
191	858970808	722382340	79995959	492084645	466643562
196	878359268	584231199	173777874	684731249	278111198

```
seed for next table:      597239251
```

付録 B　1 桁の乱数表 (seed=123456789)

XLISP-STAT において、下記のように、促進記号（>）の後に、
(table-random-number-1 123456789 'tetsuo-kamijo 123456789)
と入力することによって出力される。

```
> (table-random-number-1 123456789 'tetsuo-kamijo 123456789)

  Table of Random Number(1 digit)
    No within class:   123456789
    Name           TETSUO-KAMIJO
    Initial seed:      123456789

------------------------------------------------------------------
   i      i    i+1   i+2   i+3   i+4   i+5   i+6   i+7   i+8   i+9
------------------------------------------------------------------
   1      2     9     8     5     4     0     2     1     0     6
  11      0     4     4     7     7     0     8     3     0     0
  21      8     8     1     0     2     9     1     3     8     2
  31      6     5     8     4     9     5     1     7     3     1
  41      5     3     6     3     2     7     1     2     8     8
  51      0     7     0     7     7     3     8     7     7     4
  61      9     5     0     5     0     3     2     8     0     0
  71      1     1     3     1     2     2     6     8     5     2
  81      7     7     4     8     9     9     5     0     4     2
  91      9     6     4     1     8     9     7     4     6     4
 101      5     8     1     7     7     3     5     3     2     8
 111      7     9     1     2     1     7     3     8     6     9
 121      7     0     1     9     5     7     9     6     0     5
 131      0     6     7     7     4     1     2     8     0     7
 141      1     8     4     1     8     4     5     2     5     6
 151      0     0     6     4     6     4     2     7     7     4
 161      6     5     3     6     7     9     1     5     4     8
 171      8     0     3     4     5     3     5     7     5     3
 181      5     5     1     8     5     3     5     4     6     1
 191      8     7     0     4     4     8     5     1     6     2
 201      2     0     6     2     8     4     3     0     6     0
 211      5     0     9     3     8     7     4     9     0     8
 221      6     3     0     2     9     4     4     2     3     4
 231      9     3     3     6     4     3     4     0     8     0
 241      0     0     4     6     5     2     9     8     2     0
 251      0     5     6     3     1     6     4     2     6     6
 261      2     7     6     8     9     3     1     1     7     4
 271      2     7     8     6     6     6     8     9     5     6
 281      3     0     0     4     6     3     4     6     0     3
 291      5     9     8     4     2     3     7     0     9     0
 301      9     6     6     3     9     0     3     5     8     9
 311      7     3     1     2     7     4     5     3     7     1
 321      5     7     4     2     7     9     6     3     0     9
 331      7     2     3     3     6     9     9     3     0     5
 341      2     9     2     0     6     9     0     9     1     5
 351      3     0     2     1     1     6     8     8     0     5
 361      0     2     1     6     0     1     5     8     8     0
 371      2     8     7     7     4     9     2     9     9     3
 381      3     3     2     0     8     7     5     9     4     6
 391      1     9     6     7     0     2     4     2     5     6
------------------------------------------------------------------
seed for next table:     1349476249
```

付録 C　2 桁の乱数表 (seed=123456789)

XLISP-STAT において、下記のように、促進記号（>）の後に、
(table-random-number-2 123456789 'tetsuo-kamijo 123456789)
と入力することによって出力される。

```
> (table-random-number-2 123456789 'tetsuo-kamijo 123456789)

  Table of Random Number(2 digits)
    No within class:   123456789
    Name           TETSUO-KAMIJO
    Initial seed:      123456789

    If a digit number of random number is 1,
    then add one zero before the random number.
    For example, random number is 6, then
    the number becomes 06.
```

i	i	i+1	i+2	i+3	i+4	i+5	i+6	i+7	i+8	i+9
1	21	95	82	56	41	6	25	10	4	63
11	6	44	40	75	79	0	89	35	9	1
21	85	84	12	0	26	91	11	35	82	26
31	69	56	86	45	91	59	18	76	39	18
41	57	36	61	36	21	71	11	29	82	82
51	6	71	8	77	74	30	89	76	76	40
61	93	56	1	50	4	36	27	85	2	1
71	15	11	35	11	20	21	61	80	58	21
81	76	72	44	85	94	90	51	3	48	29
91	90	66	41	15	83	96	74	45	65	40
101	54	83	14	71	77	36	53	37	26	87
111	76	91	13	29	11	77	32	87	67	94
121	79	2	10	99	56	70	99	60	4	56
131	2	60	72	74	46	13	20	86	4	77
141	15	89	48	14	89	44	59	22	57	63
151	6	0	62	41	62	46	25	75	76	40
161	66	58	30	68	78	91	13	59	49	85
171	84	8	38	41	56	39	53	77	50	30
181	58	55	19	85	52	34	53	40	69	12
191	85	72	7	49	46	87	58	17	68	27
201	21	6	67	26	82	44	35	3	62	8
211	51	1	91	39	87	72	45	96	7	87
221	68	33	3	21	90	45	47	25	30	43
231	97	33	35	61	42	36	43	7	84	6
241	0	0	48	62	59	29	94	82	29	3
251	3	51	67	39	15	66	42	23	62	62
261	28	76	66	82	98	35	15	17	77	44
271	26	76	86	60	62	62	83	91	57	62
281	36	2	2	43	65	37	49	61	5	34
291	53	96	82	48	28	31	75	1	95	7
301	92	69	62	32	97	5	39	50	82	93
311	71	35	14	20	74	49	50	37	79	19
321	51	70	42	26	79	98	60	35	9	96
331	70	21	37	37	67	93	93	39	2	52
341	26	90	26	1	68	93	7	94	19	54
351	37	0	20	12	11	66	84	80	1	55
361	9	21	12	65	7	17	51	84	87	2
371	27	82	71	71	40	93	27	93	92	34
381	39	31	26	1	87	72	59	97	46	62
391	12	91	63	70	1	22	49	24	58	62

```
seed for next table:     1349476249
```

付録 D　3 桁の乱数表 (seed=123456789)

XLISP-STAT において、下記のように、促進記号（>）の後に、
(table-random-number-3 123456789 'tetsuo-kamijo 123456789)
と入力することによって出力される。

```
> (table-random-number-3 123456789 'tetsuo-kamijo 123456789)

  Table of Random Number(3 digits)
    No within class:   123456789
    Name           TETSUO-KAMIJO
    Initial seed:      123456789

    If a digit number of random number is n(<=2),
    then add (3-n) zero(s) before the random number.
    For example, random number is 61, then
    the number becomes 061.
```

i	i	i+1	i+2	i+3	i+4	i+5	i+6	i+7	i+8	i+9
1	218	956	829	561	415	66	257	109	43	633
11	61	449	401	754	797	1	897	350	94	13
21	859	840	123	7	260	912	113	351	822	267
31	692	561	861	453	911	597	188	761	396	185
41	574	367	617	361	212	714	117	299	825	824
51	61	710	88	777	745	308	899	763	761	406
61	938	562	17	501	41	368	271	858	29	17
71	152	114	353	119	206	212	612	809	587	215
81	768	723	448	855	945	909	519	30	481	292
91	902	667	412	156	833	964	740	456	653	406
101	540	832	145	717	775	362	531	379	269	877
111	761	913	135	291	118	771	320	877	679	949
121	793	21	102	991	563	704	994	604	46	562
131	21	609	728	748	469	135	202	860	40	773
141	155	895	487	148	893	445	599	228	572	639
151	65	2	629	412	628	469	258	754	768	403
161	660	584	304	684	780	910	131	595	490	854
171	848	83	383	411	565	394	538	775	509	305
181	584	559	195	856	529	344	530	406	692	121
191	858	722	79	492	466	878	584	173	684	278
201	214	68	679	261	829	442	351	32	628	86
211	516	11	919	396	871	722	455	966	78	870
221	688	337	36	214	908	455	474	258	302	434
231	971	335	358	615	424	366	438	78	849	65
241	3	8	486	622	598	290	944	824	290	38
251	34	510	677	397	151	662	428	236	628	621
261	281	769	663	824	986	350	151	179	770	441
271	263	761	869	600	629	623	830	919	573	620
281	368	24	23	430	652	373	493	618	55	341
291	532	969	825	485	285	319	754	14	956	75
301	927	692	629	328	975	54	393	507	822	935
311	713	358	143	207	749	490	502	373	795	192
321	511	708	428	267	791	988	601	352	92	967
331	704	215	377	379	671	937	936	394	20	522
341	264	907	267	14	687	934	74	947	193	542
351	376	9	205	128	119	665	842	806	13	557
361	95	217	121	657	75	179	519	847	875	28
371	273	824	712	713	407	933	273	931	926	340
381	394	310	264	18	876	727	597	971	469	621
391	125	916	634	708	19	228	496	246	583	628

```
seed for next table:     1349476249
```

参考文献

[1] Abelson, H., Sussman, G.J., and Sussman, J.(1996), *Structure and Interpretation of Computer Programs*, Second Edition, The MIT Press.
邦訳: 和田英一訳 (2000), 『計算機プログラムの構造と解釈』, 第 2 版, ピアソン・エデュケーション.

[2] Bratley, P., Fox, B.L., and Schrage L.E.(1987), *A Guide to Simulation*, Second Edition, Springer.

[3] Freund, J.E. and Williams, F.J.(1977), *Elementary Business Statistics: The Modern Approach*, Third Edition, Prentice-Hall.
邦訳 (Second Edition の邦訳): 福場 庸, 大澤 豊 共訳 (1974), 『経済経営系のための統計学入門』, 培風館.

[4] 藤曲哲郎, 森 隆一, 山本 浩 (1996), 『シミュレーションによる統計学』, 日本評論社.

[5] 伏見正則 (1989), 『乱数』, 東京大学出版会.

[6] Hammersley, J.M. and Handscomb, D.C.(1964), *Monte Carlo Methods*, Methuen.

[7] Harvey, B. and Wright, M.(1994), *Simply Scheme: Introducing Computer Science*, The MIT Press.

[8] IBM(1959), *Random Number Generation and Testing: Reference Manual*, IBM.

[9] 石川 宏 (1994), 『C によるシミュレーション・プログラミング』, ソフトバンク.

[10] 上條哲男 (1998), 「乱数について」, 『上智経済論集』, Vol.43, No.2, pp.139-178.

[11] 上條哲男 (2000), 「LISP および Scheme による乱数生成」, 『上智経済論集』, Vol.45, No.1, 2, pp.57-90.

[12] 上條哲男 (2004), 「LISP-STAT による無作為標本抽出のコンピュータ・シミュレーション： 有限母集団の場合 」, 『Web サイト (http://pweb.sophia.ac.jp/ ~tkamijo/)』, pp.1-16.

[13] 柏木 潤 (1996), 『M 系列とその応用』, 昭晃堂.

[14] 木下宗七 編 (1996), 『入門統計学』, 有斐閣.

[15] Knuth, D.E.(1981), *The Art of Computer Programming, Vol.2, Seminumerical Algorithms*, 2nd ed.,Addison-Wesley.
邦訳 1: 渋谷政昭訳 (1981), 『準数値算法/乱数』 (第 3 分冊), サイエンス社.
邦訳 2: 中川圭介訳 (1986), 『準数値算法/情報構造』 (第 4 分冊), サイエンス社.

[16] 栗原正仁 (1993),『対話による Common Lisp 入門』, 森北出版.

[17] Law, A.M. and Kelton, W.D.(1991), *Simulation Modeling and Analysis*, Second Edition, McGraw-Hill.

[18] Marse, K. and Roberts, S.D.(1983), "Implementing a Portable FORTRAN Uniform (0, 1) Generator," *Simulation*, Vol.41(October), pp.135-139.

[19] McCarthy, J., Abrahams, P.W., Edwards, D.J., Hart, T.P., and Levin, M.I.(1962), *LISP 1.5 Programmer's Manual*, The MIT Press.

[20] 宮武 修, 脇本和昌 (1978),『乱数とモンテカルロ法』, 森北出版.

[21] Naylor, T.H., Balintfy, J.L., Burdick, D.S. and Chu, K.(1966), *Computer Simulation Techniques*,Wiley.
邦訳: 水野幸男, 小柳佳勇 共訳 (1971),
『コンピュータ シミュレーション』, 培風館.

[22] Park, S.K. and Miller, K.W.(1988), "Random Number Generators: Good Ones Are Hard To Find," *Communications of the ACM*, Vol.31, No.10(October), pp.1192-1201.
邦訳: 西村恕彦訳 (1993),「乱数生成系で良質のものはほとんどない」,
邦訳 1: 『bit』, Vol.25, No.4(April), pp.19-27.
邦訳 2: 『bit』, Vol.25, No.5(May), pp.12-20.

[23] Pidd, M.(1998), *Computer Simulation in Management Science*, Fourth Edition, Wiley.

[24] Rubinstein, R. and Melamed, B.(1998), *Modern Simulation and Modelling*, Wiley.

[25] Soft Warehouse(1992), *muLISP 90, Reference Manual*, Soft Warehouse.

[26] Springer, G. and Friedman, D.P.(1989), *Scheme and the Art of Programming*, The MIT Press.

[27] Steele, Jr., G.L.(1990), *Common LISP: The Language*, Second Edition, Digital Press.
邦訳: 井田昌之 翻訳監修 (1991),『COMMON LISP』, 第 2 版, 共立出版.

[28] 竹村彰通 (1997),『統計』, 共立出版.

[29] 竹内郁雄 (1986),『初めての人のための LISP』, サイエンス社.

[30] 垂水共之 (1999),『Lisp-Stat による統計解析入門』, 共立出版.

[31] Tierney, L.(1990), *LISP-STAT: An Object-Oriented Environment for Statistical Computing and Dynamic Graphics*,Wiley.
邦訳: 垂水共之，鎌倉稔成，林 篤裕，奥村晴彦，水田正弘 共訳 (1996)，『LISP-STAT 入門』，共立出版.

[32] 津田孝夫 (1995),『モンテカルロ法とシミュレーション』, 三訂版，培風館.

[33] Winston, P.H. and Horn, B.K.P.(1989), *LISP*, 3rd Edition, Addison Wesley.
邦訳: 白井良明，安部憲広，井田昌之 共訳 (1996)，LISP, 原書第 3 版，( )( ).

[34] 湯浅太一 (1991),『Scheme マニュアル』, 岩波書店.

[35] 湯浅太一 (1991),『Scheme 入門』, 岩波書店.

[36] 湯浅太一, 萩谷昌己 (1986),『Common Lisp 入門』, 岩波書店.