とある授業のレポート

ひとみさん

2020 年 1 月 22 日

概要

Benford の法則とは、自然界に現れる数字を集めたときに、一番上の位の数字の出現頻度は対数的な分布をし、1 が最も出現頻度が高いという法則である。この法則に、気象計測で得た膨大な数値のデータが従っているかを確認した。

1 はじめに

Benford の法則とは、次のような法則である。

Benford の法則

自然界に出現する数値で、最上位の数字 d が n と等しくなる確率 $\Pr[d = n]$ は

$$\log_{10}\left[\frac{n+1}{n}\right]$$

この法則[1)]は会計の現場で、不正を検証するために実際に利用されている。しかしながら、Benford の法則の存在を知っていても、それを実際に確かめることができるほど多くの数値のデータを入手する機会はなかなかない。そこで、8 号館の気象データをもちいて Benford の法則を検証することとした。

2 Benford の法則とは

Benford の法則とは、自然界に出てくる多くの数値の最初の桁の分布が、一様分布ではなく対数的な分布になっているという法則である。ここで、対数的な分布 (logarithmic distribution) とは、対数目盛の各区間の広さ $\log_{10}[d+1] - \log_{10}[d] = \log_{10}[1 + 1/d]$ に比例した分布という意味である。

2.1 Benford の法則の実例

8 号館のデータについて検証する前に、Benford の法則が成り立つ実例と、その簡易な証明について記す。

2.1.1 数列 $\{2^k\}$

数列 $\{2^k\}$ に現れる数値は Benford の法則に従う。$1 \leqslant k \leqslant 100$ の範囲で、最上位の桁の数字の現れる頻度を表 1 と図 1 に示した。図を見ると、100 個という少ないサンプル数でも、非常によく Benford の法則に従っているのがわかる。

数列 $\{2^k\}$ に現れる数値は Benford の法則に従うことついての簡易な証明を以下に示す。

2^k の最上位の数字を d とし、m を整数とすると、

$$d \times 10^m \leqslant 2^k < (d+1) \times 10^m$$

常用対数を取ると、

$$m + \log[d] \leqslant k \log[2] < m + \log[d+1]$$
$$\frac{m + \log[d]}{\log[2]} \leqslant k < \frac{m + \log[d+1]}{\log[2]}$$

$d = d_n$ のとき、不等式を満たす k の幅 x_n を求める。

$$x_n = \frac{m + \log[d_n]}{\log[2]} - \frac{m + \log[d_n + 1]}{\log[2]}$$
$$= \frac{1}{\log[2]} \cdot \log\left[\frac{d_n + 1}{d_n}\right]$$

これより、x_n は m（桁数）によらない事がわかる。

次に、$\sum_{i=1}^{9}[x_i]$ を求めておく。

$$\sum_{i=1}^{9}[x_i] = \frac{1}{\log[2]} \sum_{i=1}^{9} \log\left[\frac{d_i + 1}{d_i}\right]$$

1) 実際にはこの法則を、最上位の数字だけではなく、上数桁の出現頻度へと拡張したものが使われている。

$$= \frac{1}{\log[2]} \cdot \log\left[\frac{2}{1} \cdot \frac{3}{2} \cdot \cdots \cdot \frac{10}{9}\right]$$
$$= \frac{1}{\log[2]} \cdot \log[10] = \frac{1}{\log[2]}$$

k は一様分布なので、$d = n$ となる確率 $\Pr[d = n]$ は、

$$\Pr[d = n] = \frac{x_n}{\sum_{i=1}^{9}[x_i]}$$
$$= \frac{1}{\log[2]} \cdot \log\left[\frac{n+1}{n}\right] \Big/ \frac{1}{\log[2]}$$
$$= \log\left[\frac{d+1}{d}\right]$$

2.1.2 フィボナッチ数列 $\{F_k\}$

フィボナッチ数列 $\{F_k\}$ は以下で定義される数列である。

$$\begin{cases} F_1 & = 1 \\ F_2 & = 1 \\ F_{n+2} & = F_{n+1} + F_n \end{cases}$$

フィボナッチ数列も Benford の法則に従う。フィボナッチ数列についても、$1 \leqslant k \leqslant 100$ の範囲で、最上位の桁の数字の現れる頻度を表 1 と図 1 に示した。こちらもよく Benford の法則に従っている。このことについての証明も記す。

フィボナッチ数列の漸化式から一般項を求めると、

$$F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

である。

この式の第 2 項 $\left(\frac{1-\sqrt{5}}{2}\right)^n/\sqrt{5}$ は、$n = 1$ のときに最大値 $1/\sqrt{5} \approx 0.447$ を取るので、第 2 項を省略した

$$F_n \approx \frac{\phi^n}{\sqrt{5}}, \qquad \phi = \frac{1+\sqrt{5}}{2} \text{ (黄金比)}$$

は F_n の値を 0.447 以下の誤差で、特に $n > 4$ のときは 1% 以下の誤差で与える近似式である。今は最上位の桁の数字しか着目していないので、近似的に $F_n = \phi^n/\sqrt{5}$ とする。

このように近似すると、$\{2^k\}$ と同じ議論をすることによって、同じ確率を得ることができる。

$$\Pr[d = n] = \log_{10}\left[\frac{n+1}{n}\right]$$

3 8 号館のデータを解析する

8 号館の気象データをもちいて Benford の法則を検証することとした。検証に用いたデータは、4 月 19 日から 7 月 8 日までのデータである。検証の対象は日付と時刻を除く全てのデータである。データを解析するのに用いた TEX コード、Python コード、Ruby コードを A、B、C に示した。このコードの開発には、工学院や工学部の方に協力いただいた。また、このコードで解析した結果を表 2 と図 3 に示した。TEX での実装と、python と Ruby での実装とで結果が異なるのは、TEX で処理するためにはデータの前処理が必要で、その前処理の方法が良くなかったからなのではないかと考えている。

グラフを見るだけでは、最上位桁の頻度分布が対数的になっているかわからない。特に、グラフと比較すると 1 の出現する頻度が多すぎるようにも感じられる。そこで Python のライブラリを用いて、頻度分布が対数的になっているか χ^2 検定を実施した。その結果は次のようになった。

$$\chi^2 = 68084.00878444461$$
$$p = 0.0$$

p 値は実際の分布が、期待される分布にどれだけ従っているかを表す指標である。この場合、p 値は 0 であるから帰無仮説は棄却される。したがって、8 号館のデータは Benford の法則に従っていないと言

表 1 $1 \leqslant n \leqslant 100$ の範囲における最上位桁の出現頻度

数字	2^n	F_n
1	30	30
2	17	18
3	13	13
4	10	9
5	7	8
6	7	6
7	6	5
8	5	7
9	5	4

える。

3.1 気温データに限定して検証する

8 号館のデータを解析しても、Benford の法則に従っていなかった理由を考えたところ、次の点が原因なのではないかと考えた。それは、解析の対象となるデータの中に気圧のデータが含まれており、気圧の値の最上位桁は殆どの場合 1、まれに 9 になるのみで、明らかに Benford の法則に従わないため、それの影響をうけているのではないかということだ。

そこで、次に気温を表すデータのみに絞って、Python で解析を行った。その結果を表 3 と図 3.1 に示す。また、先と同じように χ^2 検定も行った。その結果は次のようである。

$$\chi^2 = 24282.84808321144$$
$$p = 0.0$$

こちらも p 値が 0 なので、気温データのみで解析しても、Benford の法則に従っているとは言えなかった。

4 まとめ

8 号館から得た気象データは、Benfoed の法則に従っているとは言えないことがわかった。逆に言えば、8 号館から得た気象データは、Benford の法則に従うような確率分布をもっていないということである。

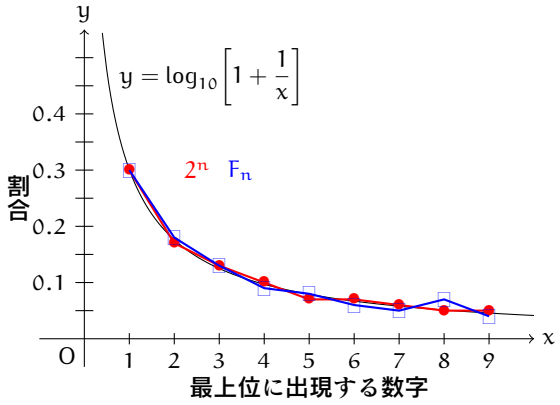

図 1　$1 \leqslant n \leqslant 100$ の範囲における最上位桁の出現頻度

Benford の法則に従うような確率分布は、スケール不変な確率分布であるということが知られている。スケール不変とは、観測対象 F について、任意のスケール変換 $x \to \lambda x$ に対し、

$$F(\lambda x) = \mu F(x)$$

を満たす定数 μ が存在するような F のことである。スケール不変なものの例として、べき乗則 $F(x) = cx^{\alpha}$ があげられる。8 号館の気象データが Benford の法則に従っていないということは、8 号館の気象データはスケール不変ではないということでもある。

今回は 4 月から 7 月までのデータでしか解析を行わなかったが、通年のデータで解析を行ったらどのような結果になるのかの検証も待たれる。

5 参考文献

- http://zakii.la.coocan.jp/enumeration/64_benford.htm
- https://ja.wikipedia.org/w/index.php?title=%E3%83%99%E3%83%B3%E3%83%95%E3%82%A9%E3%83%BC%E3%83%89%E3%81%AE%E6%B3%95%E5%89%87&oldid=68691651
- https://ja.wikipedia.org/w/index.php?title=%E3%82%B9%E3%82%B1%E3%83%BC%E3%83%AB%E4%B8%8D%E5%A4%89%E6%80%A7&oldid=63430785

表 2　8 号館のデータに現れる最上位桁の出現頻度

数字	TEX	割合 [%]	Python	Ruby	割合 [%]
1	133358	49.9	139375	139375	50.028
2	52413	18.8	46462	46462	16.677
3	8957	3.2	8008	8008	3.162
4	12325	4.4	13558	13558	4.867
5	14052	5.0	15496	15496	5.562
6	13366	4.8	12263	12263	4.402
7	12939	4.6	13197	13197	4.737
8	15063	5.4	14403	14403	5.170
9	15814	5.6	15034	15034	5.396
計	278278	100.0	278596	278596	100.000

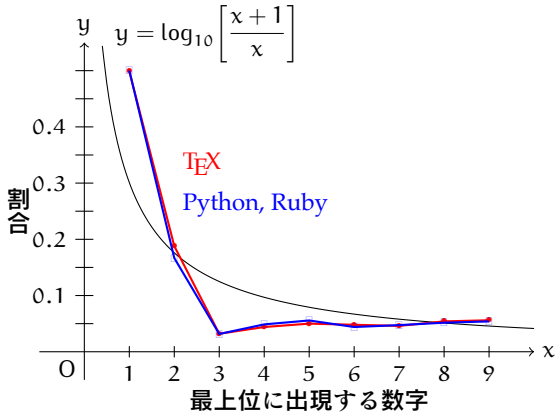

図 2　8 号館のデータに現れる最上位桁の出現頻度

表 3　8 号館の気温データに現れる最上位桁の出現頻度

数字	度数	割合 [%]
1	22122	63.22
2	5817	16.62
3	94	0.27
4	99	0.28
5	301	0.86
6	550	1.57
7	1742	4.98
8	2266	6.48
9	2271	6.49
計	34992	100.00

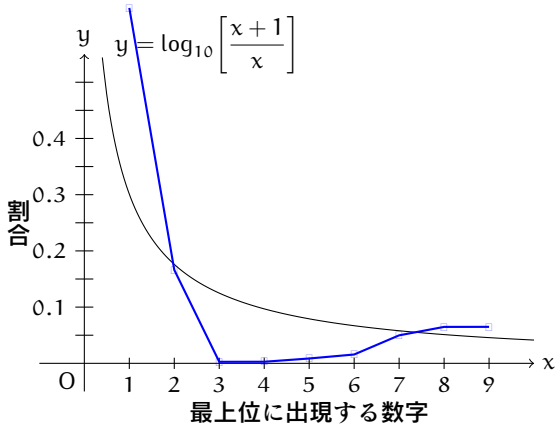

図 3　8 号館の気温データに現れる最上位桁の出現頻度

A　8 号館のデータを解析するのに用いた TEX コード

この TEX ソースをコンパイルするために、気象データのファイルを表計算ソフトなどで処理し、以下の要件を満たすようにファイルを編集する必要がある。なお、そのファイル名は data2.tex とする。

- 解析の対象としない、日付や時刻をあらわす列を削除する。
- 全ての数値を指数表記にする。
- コンマ区切り（CSV 形式）にする。
- 全ての行頭に\counttop:と書く。

Listing 1　test.tex

```
\documentclass[uplatex,8pt]{jsarticle}
\usepackage{counttop}
\begin{document}
\catcode`,=13
\def,{\counttop:}
\include{data2}
\countshow
\end{document}
```

Listing 2　counttop.sty

```
% 1 14 514 1919 810 334 のような数値のリストが有って、
% \counttop1 \counttop14 \counttop514 ...のように
% することで、最上位の数字をカウントするLaTeXプログラム?

\newcounter{hm@count@one}\setcounter{hm@count@one}{0}
\newcounter{hm@count@two}\setcounter{hm@count@two}{0}
\newcounter{hm@count@three}\setcounter{hm@count@three}{0}
\newcounter{hm@count@four}\setcounter{hm@count@four}{0}
\newcounter{hm@count@five}\setcounter{hm@count@five}{0}
\newcounter{hm@count@six}\setcounter{hm@count@six}{0}
\newcounter{hm@count@seven}\setcounter{hm@count@seven}{0}
\newcounter{hm@count@eight}\setcounter{hm@count@eight}{0}
\newcounter{hm@count@nine}\setcounter{hm@count@nine}{0}

\def\counttop:#1{%
  \ifx#11\relax\stepcounter{hm@count@one}\else
    \ifx#12\relax\stepcounter{hm@count@two}\else
      \ifx#13\relax\stepcounter{hm@count@three}\else
        \ifx#14\relax\stepcounter{hm@count@four}\else
          \ifx#15\relax\stepcounter{hm@count@five}\else
            \ifx#16\relax\stepcounter{hm@count@six}\else
              \ifx#17\relax\stepcounter{hm@count@seven}\else
                \ifx#18\relax\stepcounter{hm@count@eight}\else
                  \ifx#19\relax\stepcounter{hm@count@nine}\else
                    \relax\fi\fi\fi\fi\fi\fi\fi\fi\fi\relax#1}

\def\countshow{%
\newcounter{hm@count@sum}\setcounter{hm@count@sum}{0}
\addtocounter{hm@count@sum}{\value{hm@count@one}}
\addtocounter{hm@count@sum}{\value{hm@count@two}}
\addtocounter{hm@count@sum}{\value{hm@count@three}}
\addtocounter{hm@count@sum}{\value{hm@count@four}}
\addtocounter{hm@count@sum}{\value{hm@count@five}}
```

```
\addtocounter{hm@count@sum}{\value{hm@count@six}}
\addtocounter{hm@count@sum}{\value{hm@count@seven}}
\addtocounter{hm@count@sum}{\value{hm@count@eight}}
\addtocounter{hm@count@sum}{\value{hm@count@nine}}
\textbf{\textsf{RESULT}}:\\
\textbf1: \the\value{hm@count@one}\hfill
\textbf2: \the\value{hm@count@two}\hfill
\textbf3: \the\value{hm@count@three}\hfill
\textbf4: \the\value{hm@count@four}\hfill
\textbf5: \the\value{hm@count@five}\hfill
\textbf6: \the\value{hm@count@six}\hfill
\textbf7: \the\value{hm@count@seven}\hfill
\textbf8: \the\value{hm@count@eight}\hfill
\textbf9: \the\value{hm@count@nine}\\
\textbf{TOTAL}: \the\value{hm@count@sum}\par
\newcounter{hm@temp}
\setcounter{hm@temp}{\value{hm@count@one}}\multiply \c@hm@temp 1000
\newcounter{hm@count@temp@one}\setcounter{hm@count@temp@one}{\value{hm@temp}}
\setcounter{hm@temp}{\value{hm@count@two}}\multiply \c@hm@temp 1000
\newcounter{hm@count@temp@two}\setcounter{hm@count@temp@two}{\value{hm@temp}}
\setcounter{hm@temp}{\value{hm@count@three}}\multiply \c@hm@temp 1000
\newcounter{hm@count@temp@three}\setcounter{hm@count@temp@three}{\value{hm@temp}}
\setcounter{hm@temp}{\value{hm@count@four}}\multiply \c@hm@temp 1000
\newcounter{hm@count@temp@four}\setcounter{hm@count@temp@four}{\value{hm@temp}}
\setcounter{hm@temp}{\value{hm@count@five}}\multiply \c@hm@temp 1000
\newcounter{hm@count@temp@five}\setcounter{hm@count@temp@five}{\value{hm@temp}}
\setcounter{hm@temp}{\value{hm@count@six}}\multiply \c@hm@temp 1000
\newcounter{hm@count@temp@six}\setcounter{hm@count@temp@six}{\value{hm@temp}}
\setcounter{hm@temp}{\value{hm@count@seven}}\multiply \c@hm@temp 1000
\newcounter{hm@count@temp@seven}\setcounter{hm@count@temp@seven}{\value{hm@temp}}
\setcounter{hm@temp}{\value{hm@count@eight}}\multiply \c@hm@temp 1000
\newcounter{hm@count@temp@eight}\setcounter{hm@count@temp@eight}{\value{hm@temp}}
\setcounter{hm@temp}{\value{hm@count@nine}}\multiply \c@hm@temp 1000
\newcounter{hm@count@temp@nine}\setcounter{hm@count@temp@nine}{\value{hm@temp}}
\divide \c@hm@count@temp@one \c@hm@count@sum
\divide \c@hm@count@temp@two\c@hm@count@sum
\divide \c@hm@count@temp@three \c@hm@count@sum
\divide \c@hm@count@temp@four \c@hm@count@sum
\divide \c@hm@count@temp@five \c@hm@count@sum
\divide \c@hm@count@temp@six \c@hm@count@sum
\divide \c@hm@count@temp@seven \c@hm@count@sum
\divide \c@hm@count@temp@eight \c@hm@count@sum
\divide \c@hm@count@temp@nine \c@hm@count@sum
\textbf{\textsf{RATIO}}:\\
\textbf1: \the\value{hm@count@temp@one}\hfill
\textbf2: \the\value{hm@count@temp@two}\hfill
\textbf3: \the\value{hm@count@temp@three}\hfill
\textbf4: \the\value{hm@count@temp@four}\hfill
\textbf5: \the\value{hm@count@temp@five}\hfill
\textbf6: \the\value{hm@count@temp@six}\hfill
\textbf7: \the\value{hm@count@temp@seven}\hfill
\textbf8: \the\value{hm@count@temp@eight}\hfill
\textbf9: \the\value{hm@count@temp@nine}\\
}
```

B　8 号館のデータを解析するのに用いた python コード

```
import csv
import re
import numpy
from scipy.stats import chisquare

# 正規表現
reReal = re.compile('^[+-]?[0-9]+(\.[0-9]+)?$')

# カウント用のハッシュ
counter = {}
for i in range(1, 10):
        counter[str(i)] = 0

# ファイルを捜査する
with open('20180419_0708.txt', 'r') as f:
        reader = csv.reader(f, delimiter='\t')
        for row in reader:
                # print(row)
                for item in row:
                        # 実数のフォーマットに従っているか
                        if reReal.search(item):
                                # print(item)
                                # 1-9の文字以外を捨てる
                                item = re.sub('[\D0]', '', item)
                                if item == '':
                                        item = '0'
                                initialNum = item[0]
                                # 有効数字の最上位が0のときは無視して次へ
                                if initialNum == '0':
                                        continue
                                counter[initialNum] += 1

print(counter)

# サンプル数
nSample = sum(counter.values())
print(nSample)

# 理想と実際を比較
actual = counter.values()
expected = []

for i in range(1, 10):
        expected.append(nSample * numpy.log10((i+1)/i))
print(expected)

# タプルに変換
actual = tuple(actual)
expected = tuple(expected)

# print(len(actual))
# print(len(expected))

# 検定
chisq, p = chisquare(actual, f_exp = expected)
```

```
print('X^2 = ', chisq)
print('p = ', p)
```

C　8 号館のデータを解析するのに用いた Ruby コード

```
require "csv"


# メソッドの定義
# 引数の上1桁を返す。文字列、空白、0の場合0を返す
def first_num(number)
        if(number==nil)
                return 0
        end

        number = number.delete("^0-9")
        number = number.to_i
        while number>9
                number = number/10
        end
        return number
end


# 読み込むファイルの指定
# tab区切りTSVのみ対応。
# 同一ディレクトリに入ったファイルを読む気がする
arr = CSV.read('20180419_0708.txt','r:UTF-8',col_sep:"\t")

# 個数配列
n_num = [0,0,0,0,0,0,0,0,0]

# 個数をカウント
for i in 2..arr.length-1
        for j in 2..arr[2].length-1
                n_num[first_num(arr[i][j])]+=1
        end
end

# 合計…0は足さない
sum=0
for i in 1..9
        sum+=n_num[i]
end

# 割合
r_num=[]
for i in 1..9
        r_num[i]=n_num[i].to_f/sum*100
end

# ベンフォードの法則
ben=[]
for i in 1..9
        ben[i]=Math.log10(1.0+1.0/i)*100
end

# 出力
print "   集計した数値：      N個：   割合 ：  参考\n"
for i in 1..9
```

```
        printf("%dから始まる数値：%6d個：%6.3f%%: %6.3f%%\n",i,n_num[i],r_num[i],ben[i])
end
```