Benford's Law 在编程数据中的简单探讨

《无处不在的数学》期中作业论文

杨志斌 计算机科学与技术系 **181860126**

关于**Benford's Law** 的中文叙述：

本福特定律，也称为本福特法则，说明一堆从实际生活中的得到的数据中，以1为首位的数字出现的概率约为总数的三成，接近直觉得出的期望值$\frac{1}{9}$的三倍。推广来说，越大的数以他为首几位的数，出现的概率就越低。它可用于检查各种数据是否有造假。

在数学上，本福特定律说明了在b进制中，以数n开头的数出现的概率是$log_b(n+1)-log_b n$。本福特定律不但适用于个位数字，连多位数字也可应用。

在十进制首位数字出现的概率为：[1]

d	p
1	30.1%
2	17.6%
3	12.5%
4	9.7%
5	7.9%
6	6.7%
7	5.8%
8	5.1%
9	4.6%

关于定理的简单思考：

首先，既然是概率问题，那么可以肯定一定是在数据样本较大时，才有所体现。所以很自然的我们会想到利用计算机生成随机数并统计首位数字出现的概率。那么我设计了一个简单的c++函数来实现这一猜想：[^2]

```
#include<iostream>
#include<cstdlib>
```

```cpp
#include<ctime>
using namespace std;

//用while循环获取每个随机数的最高位数
int getHigh(int n){
    int high = -1;
    while(n>0){
        high = n % 10;
        n /= 10;
    }
return high;
}

//用一维数组记录每个随机数首位数字出现次数
//数组下标的1~9分别表示首位数字1~9
int a[10];

int main(){
    srand(time(NULL));//用c语言自带的随机数生成器产生随机数种子

    for(int i = 1; i <= 9; i++)
        a[i] = 0;//程序开始前，我们先把1~9的出现次数初始化为零

    freopen("output.txt", "a", stdout);//重定向输出到output.txt文件记录结果

    for(int i = 1; i <= 9999999; i++)//循环产生9999999个随机数，并记录
    {
        int tempInt = rand()%10000000+1;
        int tempHigh = getHigh(tempInt);
        a[tempHigh]++;//每次都把首位数字对应的数组元素加一
        cout << tempInt << "\t";//输出随机数本身
        if(i % 100 == 0){
            cout << "\n\n\n";
            for(int j = 1; j <= 9; j++)
                cout << a[j] << "\t" ;//每产生一百个随机数就输出一次首位数字统计
            cout << "\n\n\n";
        }
    }
    fclose(stdout);

return 0;
}
```

这个实验的输出数据过大，仅仅是数字存在txt文件中就达到了87M，所以不能附上详细数据，读者可以根据笔者列出的源代码自行生成数据观察（注意，因为生成的是随机数，所以具体统计数据可能会出现偏差，但数据比例不会变，所以对结论并没有影响）。这里简单描述结果：

> 数据量达到100时，从1~9的首位数字出现量分别是：
>
> 8 10 14 10 13 15 14 10 6
>
> 数据量达到1000时，从1~9的首位数字出现量分别是：

> 105 108 111 87 109 103 138 131 108
>
> 数据量达到10000时，从1~9的首位数字出现量分别是：
>
> 1083 1171 1113 1054 1126 1058 1137 1129 1129
>
> 数据量达到100000时，从1~9的首位数字出现量分别是：
>
> 10974 11196 10973 11061 11252 11193 11097 11146 11108
>
> 数据量达到1000000时，从1~9的首位数字出现量分别是：
>
> 111374 110847 110683 111024 111568 111386 111496 111034 110588

事实上不需要再列出下面的数据，我们也可以很轻松的看出这一结果并不符合本福特定律所描述的出现概率，这一结果使我感到疑惑，为了找出答案，我进行了进一步的探索。

首先当然是分析c语言自带的随机数产生机理，同时注意到本福特定律中提到了**“从实际生活中所得数据”**，首先分析是不是自己产生的随机数有问题。其次考虑到本福特定律最常见的应用是检查财务数据是否存在人为造假，所以上述的数据并不是来源于生活，这又是一个不符合定率的可能原因。第三，老师在课上曾介绍了这一定律的简单理解：随着自然数字的增长，每增加一个数位，在首位数字不变的情况下所需产生的变化越来越艰难，也就是变化所经历的数字越来越多，那么以较小数字开头的数字出现几率也就越大。（显而易见，这一证明是不严谨的，后文会加入严谨证明的简单分析）那么随机产生的数字并不经过这一数字增长过程，所以可能不遵守这一定律。

以下分析并进行实验：

我们不难找到c语言提供的$rand()$函数的实现机理：

```c
static unsigned long int next = 1;

int rand(){
    next = next * 1103515245 + 12345;
    return (unsigned int)  (next / 65535) % 32768;
}

void srand(unsigned int seed){
    next = seed;
}
```

由以上代码可以看出，c的随机数生成器产生的并不是“真正的随机数”，而是由确定的算法产生的一个数字。只有当每次都$seed$不同时，才会产生不同的数字。（而我们在程序中给出的$time()$函数则恰好满足了这一要求）

也就是说，我们的随机数“并不随机”！

但是，在我的探索过程中，偶然间发现这一程序再Windows系统下的运行结果竟然是截然不同的！

在我从机房回到宿舍时，我使用了我的笔记本上的Windows系统重新测试了这一程序，并得到了如下结果：

> 数据量达到100时，从1~9的首位数字出现量分别是：
>
> 38 32 18 0 4 3 1 2 2

> 数据量达到1000时，从1~9的首位数字出现量分别是：
>
> 335 325 153 30 32 25 34 34 32
>
> 数据量达到10000时，从1~9的首位数字出现量分别是：
>
> 3382 3423 1177 333 337 305 339 352 352
>
> 数据量达到100000时，从1~9的首位数字出现量分别是：
>
> 33789 34079 11791 3358 3382 3438 3295 3428 3440

从以上数据不难看出，这些随机数是基本遵守本福特定律的，这使我产生了极大的探索兴趣，同一段代码的结果居然如此不同，这与本福特定律本身的神秘属性如此的吻合。（从网络和课堂资料来看，直到今天也没有一份数学意义上严格的理论证明可以说明Benford's Law 的正确性（尽管这一定律已经被大数据和生活中的无数事实证明）

经过多次尝试，我证实了在Linux系统和Windows系统上确实结果有所不同，但是在网络上并没有找到相关的原因，但是既然函数原型是相同的（上文已给出）那么理论上数据的生成方法就是相同的。也就是说与上文提到的"伪随机数"无关。那么

我们再来看一看Benford's Law 的数学证明是否对我们有所帮助，经查阅网络资料，公认的对于本福特定律的理论证明是：*A Statistical Derivation of the Significant-Digit Law* [2] 但是我初步阅读一下发现，这一论文所述更像是用结果解释原因，所以我并不觉得他可以说明问题，至于进一步的原因解释，我猜想是由于不同系统上编译器对于srand(time(NULL))这一语句的解释不同造成的，所以我又查看了time函数。本身time()中参数为NULL时，函数返回从1970-1-1 0：00：00到程序运行到此语句时所经历的秒数，那么依次分析，虽然不同时间time返回值不同，但对于数据研究应该没有影响。[3]

对于这一原因，我目前想不到什么合理的解释了，在以后我继续学习计算机系统和编译原理的相关课程时，可能还会有所发现，那么先让这一奇怪的问题埋下一颗求索的种子吧！

> 作者：杨志斌 学号：181860126
>
> 邮箱：2500223255@qq.com

> 如果老师有什么解释、发现或者对本文有什么问题和建议，欢迎与我交流！

---

1. 以上段落均引用自维基百科（Wikipedia），https://zh.m.wikipedia.org/wiki/本福特定律↩

2. *By. Theodore P. Hill* ↩

3. 这是显然的，因为只要数据的种子是确定的（不相同，因为我们本身就要得到随机数）那么后续过程就不应该对首位数字的出现频率有这么大的影响↩