The University of Franche-Comté

Master CompuPhys

Statistical physics home assignment : Benford's law

Chiari Even
M1 Compuphys 2022-2023

Supervisor : Jose Lages

January 15, 2023

Contents

I Introduction

In this home assignment we will study the Benford's law define like :

$$P(i) = \log(i+1) - \log(i) = \log(\frac{i+1}{i}) = \log(1+\frac{1}{i})$$

with $i \in \{1, .., 9\}$

obviously the total probability should be equal to one. In the other basis the law is define like :

$$P(i) = \log_b(i+1) - \log_b(i) = \log_b(\frac{i+1}{i}) = \log_b(1+\frac{1}{i})$$

with $i \in$ various digit in other basis

we can easily compute this formula is base 10 with the change of base formula of log :

$$\log_b(M) = \frac{\log_x(M)}{\log_x(b)}$$

II Question 5

Now we want to check if the Fibonacci numbers follow the Benford's law, in order to do that we have to to the distribution of the first digit for Fibonacci numbers in base 10 and 16:

First i will code the a Fibonacci function to generate a list of them :

```
def fibo(n):
    """Function for fibonnaci numbers

    Args:
        n (integer): n fibonnaci numbers

    Returns:
        fibo (list) : list of n fibonnaci numbers
    """
    list=[0,1]
    for i in range(n):
        list.append(list[-2] +list[-1])
    return list
```

Now in order to understand hexadecimal number i will code two function one to convert decimal into hexadecimal and the second to convert hexadecimal into decimal :

```
def CDH(n):
    """Function to convert n in base 10 to n in base 16
        CDH for Convert Decimal to Hexadecimal

    Args:
        n (integer): number that we want to convert

    Returns:
        r (str): return n in hexadecimal so in str
    """
    r=""
    d={0:"0",1:"1",2:"2",3:"3",4:"4",5:"5",6:"6",7:"7",8:"8",9:"9",
        10:"A",11:"B",12:"C",13:"D",14:"E",15:"F"}
    while True :
        r += d[n%16]
        n = n // 16
        if n == 0 :
            return r[::-1]


def CHD(n):
    """Function to convert n in base 16 to n in base 10
        CHD for Convert Hexadecimal to Decimal

```

```
    Args:
        n (str): number that we want to convert

    Returns:
        r (integer): return n in decimal so in integer
    """
    d={0:"0",1:"1",2:"2",3:"3",4:"4",5:"5",6:"6",7:"7",8:"8",9:"9",
        10:"A",11:"B",12:"C",13:"D",14:"E",15:"F"}
    d= {v: k for k, v in d.items()}
    r, it = 0, 0
    n=n[::-1]
    for i in n:
        r+= d[i] * 16**it
        it +=1
    return r
```

In python hexadecimal number are like $0x----$ but my function generate these number like $----$ like that its easier to get the first digit of a hexadecimal number.

To see if the Fibonacci numbers follow the Benford's law i will plot in the same figure the frequency of each digit for Fibonacci numbers and Benford's law.

The distribution of each digit for benford's law are stored with a dictionary and compute like that in base 10 and in base 16 (with the change of base formula) :

```
#compute the Benford's law in base 10
#create dictionnary to have "i_digit" : frequency
it=1 #start at one because we don't take in account 0
ben_d={"1":0,"2":0,"3":0,"4":0,"5":0,"6":0,"7":0,"8":0,"9":0} #ben_d for Benford's_law_Decimal
for i in ben_d.keys():
    ben_d[i] = np.log10(1+1/it) #add np.log10(1+1/it) for Benford's law in base 10 with it the
    i_digit
    it += 1

#compute the Benford's law in base 16
#create dictionnary to have "i_digit" : frequency
it=1 #start at one because we don't take in account 0
ben_h={"1":0,"2":0,"3":0,"4":0,"5":0,"6":0,"7":0,"8":0,"9":0,
        "A":0,"B":0,"C":0,"D":0,"E":0,"F":0} #ben_h for Benford's_law_Hexadecimal
for i in ben_h.keys():
    ben_h[i] = np.log10(1+1/it)/np.log10(16) #add np.log10(1+1/it)/np.log10(16)
                                             #for Benford's law in base 16 with it the i_digit
    it += 1
```

and for the fibonacci number :

```
data=fibo(N)
for i in range(1,N):#loop over the number of elements
    nd=str(data[i]) #convert the number into str to get a list and have the first digit by list[0]
    in base 10
    nh=CDH(data[i]) #convert the number into str to get a list and have the first digit by list[0]
    in base 16
    fd_fib_d[nd[0]] = fd_fib_d.get(nd[0]) + 1/N #update the values for i digit by 1/N (already
    normalised) in base 10
    fd_fib_h[nh[0]] = fd_fib_h.get(nh[0]) + 1/N #update the values for i digit by 1/N (already
    normalised) in base 16
```

Now we can plot the histogram for the base 10 :

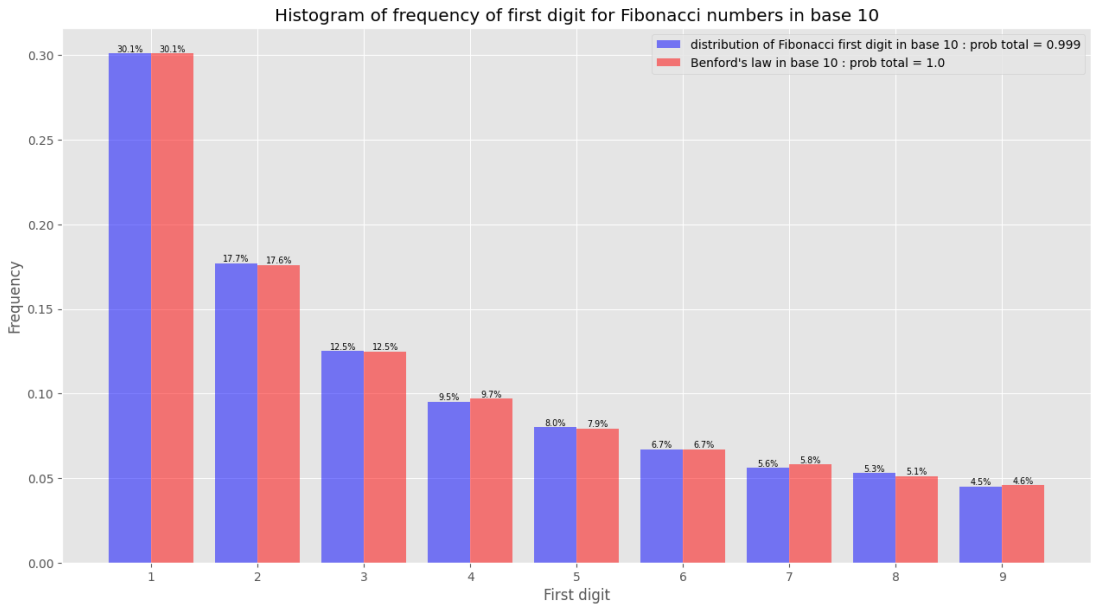

Figure 1: Histogram of distribution of the first digit for Fibonacci numbers in base 10

On this graph we can see clearly the Fibonacci numbers are following the Benford's law in base 10, the total probability should be one for any data set but for the Fibonacci numbers i have 0.999 this come from the precision of a programming language and float number because when i compute the distribution I normalise by the length of the Fibonacci list, here the number I took 1000 numbers to have enough data obviously with a number N of Fibonacci numbers too few we will not see the correlation between the two law.

We can check the correlation in base 16 :

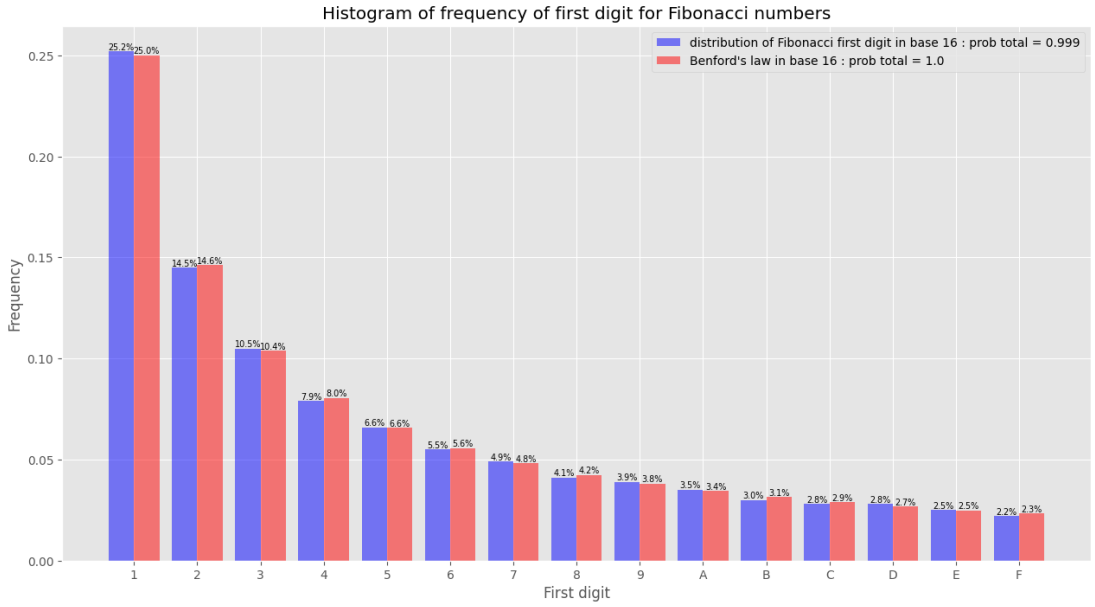

Figure 2: Histogram of distribution of the first digit for Fibonacci numbers in base 16

Even is base 16 we have a good behavior of the distribution of the first digit for Fibonacci numbers, we have the same problem of probability for the Fibonacci numbers but the probability is very close to one so we can say the correlation is correct.

III Question 6

Now we will check a correlation between two data set and the Benford's law. I am very surprising by this law so i wanted to find data set very tricky so i took the numbers of bar per country and the numbers of bar per city in US.
I have found the two data set here :

- bar per city : click here
- bar per country : click here

To obtains the number of bar per country I have code that :

```
#unpack the data from data_beer.csv
path=os.path.join(os.path.dirname(__file__),"data_beer.csv") #path of csv
df=pd.read_csv(path) #unpack with pandas
nb_bar=df['country'].value_counts() #take the colomun that we want
nb_bar=nb_bar.to_dict() #transform datafram into dictionary
```

To obtains the distribution of the first digit of the number of bar I have proceed like before :

```
#compute the distribution of first digit for bar per country in base 10 and 16
#create dictionnary to have "i_digit" : frequency
fd_bar_d={"1":0,"2":0,"3":0,"4":0,"5":0,"6":0,"7":0,"8":0,"9":0} #fd_bar_d for
    First_Digit_Bar_Decimal
fd_bar_h={"1":0,"2":0,"3":0,"4":0,"5":0,"6":0,"7":0,"8":0,"9":0,
            "A":0,"B":0,"C":0,"D":0,"E":0,"F":0} #fd_bar_h for First_Digit_Bar_Hexadecimal
```

Now we can plot the histogram in base 10 :

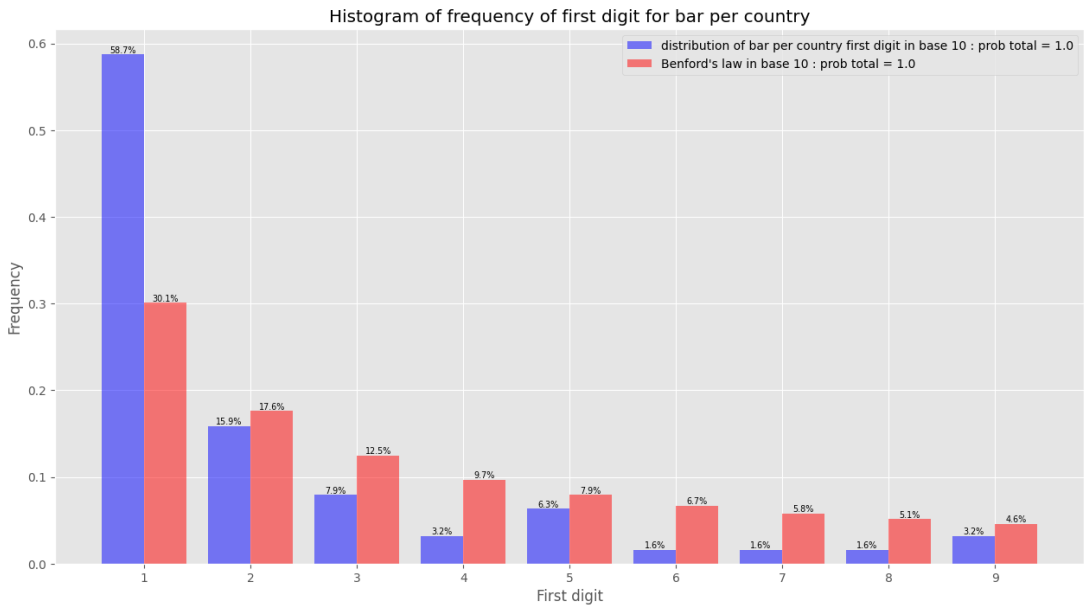

Figure 3: Histogram of distribution of the first digit for bar per country in base 10

I base 10 we can see the number of bar per country has the same tendency of benford's law but the value of distribution are not the same, they are quite close if we forgot the number one.

The histogram in base 16 :

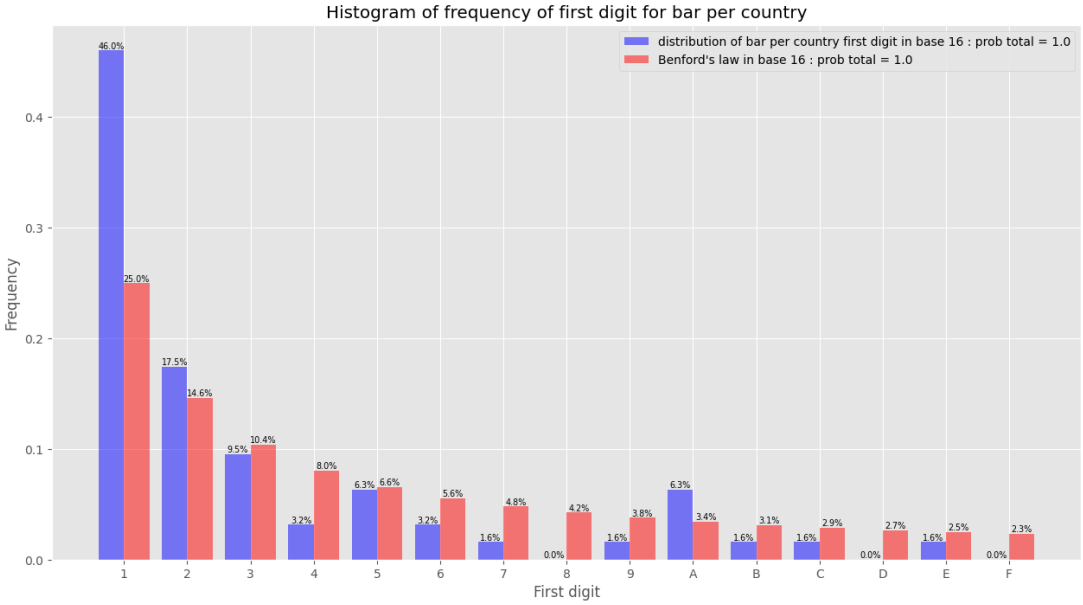

Figure 4: Histogram of distribution of the first digit for bar per country in base 16

In base 16 the tendency between the number of bar per country and the Benford's law is visible but we have like in the previous graph the values quite close for all number but not for 1 and 16.

For the second data set i have choose the city in US, to count i did like for the first data set :

```
#compute the distribution of first digit for bar per city in base 10 and 16
#create dictionnary to have "i_digit" : frequency
fd_bar_d={"1":0,"2":0,"3":0,"4":0,"5":0,"6":0,"7":0,"8":0,"9":0} #fd_bar_d for
    First_Digit_Bar_Decimal
fd_bar_h={"1":0,"2":0,"3":0,"4":0,"5":0,"6":0,"7":0,"8":0,"9":0,
          "A":0,"B":0,"C":0,"D":0,"E":0,"F":0} #fd_bar_h for First_Digit_Bar_Hexadecimal

for i in nb_bar.keys(): #loop over the number of elements
    nd=str(nb_bar[i]) #convert the number into str to get a list and have the first digit by list[0]
     in base 10
    nh=CDH(nb_bar[i]) #convert the number into str to get a list and have the first digit by list[0]
     in base 16
    fd_bar_d[nd[0]] = fd_bar_d.get(nd[0]) + 1/len(nb_bar) #update the values for i digit by 1/N (
    already normalised) in base 10
    fd_bar_h[nh[0]] = fd_bar_h.get(nh[0]) + 1/len(nb_bar) #update the values for i digit by 1/N (
    already normalised) in base 16
```

Now we can plot the histogram in base 10 :

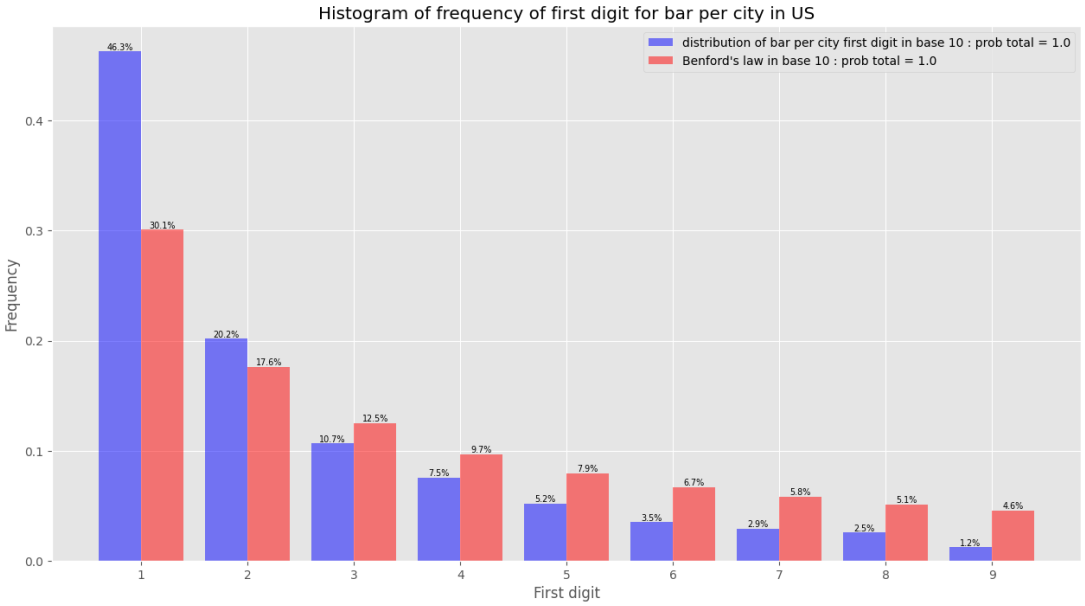

Figure 5: Histogram of distribution of the first digit for bar per city in base 10

In base 10 the number of bar per city has the same tendency of Benford's law and the same probability if we forgot the number one. In this case like in the Benford's law the probability of 1 is greater than the probability of 2 and this probability of 2 is greater than the probability in 3 and so on until 9.

Now we can plot the histogram in base 16 :

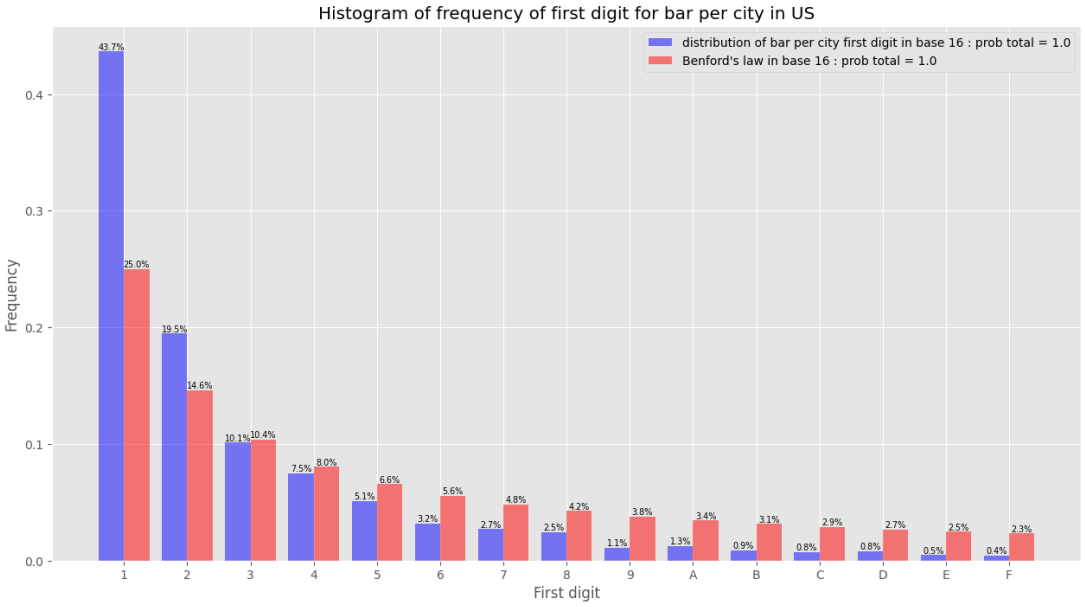

Figure 6: Histogram of distribution of the first digit for bar per city in base 16

Even in base 16 we have the same behavior of number of bar per city like in the previous graph and the same quite close probability but for the number A, B, C, D, E, F their probability are less than the other.

IV Conclusion

The Benford's law is surprising, we have check the correlation of it and Fibonacci numbers in base 10 and 16 and we have found very close probability so we can say the Fibonacci numbers follow the Benford's law. For the data set the correlation is more nuanced we have a good tendency but the probability is not exactly the same, we can say the second data set (number of bar per city in US) follow more the Benford's law than the first (number of bar per country) and this can be the consequence of the length of them because the first data set is much more larger (around 7000 bar) than the second (around 1400 bar). The probability of one is much important in the data set because the probability to have one bar is more important especially in small town.
With this conclusion we have more information to find a data set that will follows the Benford's law :

- data set with a big quantity of values better
- the data set must not have artificial minimum and maximum or have a big standard deviation
- if the data set follows the Benford's law it should be follows in base 16 (maybe in all other basis ? (obviously in base 1 and 2 because all numbers start with one))