模擬與統計計算 hw7

學號:N26120820　　姓名:羅文璟

題目

參照"TALENT VERSUS LUCK: THE ROLE OF RANDOMNESSIN SUCCESS AND FAILURE"這篇論文
設計自己的模擬世界，討論 talent, luck 與 capital/success 的關係
請在報告中說明清楚模擬世界的運作規則

規則:
模擬 10000 人，世界地圖大小為 500x500 總共 25000 個方格，每個方格上都可以站人，並且人的位置不重疊，同時世界底下存在著好事和壞事的地圖，同樣為是在 25000 個方格中，並且隨機撒點，好事和壞事可以重疊，每半年記錄一次財富，紀錄的方式是在每個人的座標周圍九宮格內的好事總數量和壞事總數量，如果半年內有 n 件壞事，那財富就會直接變為2^{-n}倍，如果有 n 件好事則會依據他的天賦決定是否把財富加倍，最多變為2^{n}倍，半年統計完後將好事和壞事隨機移動位置，移動方式為以該事件為中心，可以移動範圍為 21x21 方格內的座標，如果超過世界邊界，就會往世界另一邊的邊界移動。移動完後再重新計算一次，總共紀錄 40 年，並 print 出財富的分布，和最有錢跟最沒錢的人的財富變化。

程式碼:
定義三個 class: Person、People、event_map

```
class  Person:
    def  __init__(self):
        self.talent=0
        self.captial=10
        self.history=[10]
        self.pos=[]
        self.good=[]
        self.bad=[]
    def  printHistory(self):
        x=len(self.history)
        plt.plot(range(x),self.history)
        plt.yscale("log")
        plt.show()
    def  printEvents(self):
        plt.plot(self.good,color="red")
        plt.plot(self.bad,color="red")
        plt.show()
```

Person class 主要用來記錄一個人的 talent、captial、所在位置、歷年財富、發生的好事數量和壞事數量

下面兩個 function 則用來繪製一個人歷年財富的圖和遇到好事壞事的圖

People class 則是用來記錄所有人的財富分布、talent 分布(使用 normal distribution)，還有生成每個人的位置

```
class  People:
    def  __init__(self,n,mapSize):
        self.person=self.generatePerson(n)
        self.position=self.generatePos(n,mapSize)
        self.captial=[]
    def  generatePerson(self,n):  #生成人物和智慧
        person=[]
        for  i  in  range(n):
            person.append(Person())
        talent=np.random.normal(0.5,  0.1,  n)  #mean=0.5  std=0.15
        for  i  in  range(n):
            if(talent[i]>=1):
                talent[i]=0.99
            elif(talent[i]<0):
                talent[i]=0
            person[i].talent=talent[i]
        return  person
```

```
  def  generatePos(self,n,mapSize):  #生成人物位置，因為要不重複所以需要一次生成
      indices  =  np.random.choice(mapSize*mapSize,  n,  replace=False)
      x_indices,  y_indices  =  np.unravel_index(indices,  (mapSize,mapSize))
      pos=  list(zip(x_indices,  y_indices))
      for  i  in  range(n):
          self.person[i].pos=pos[i]
  def  CaptialDistribution(self,show=False):#畫出財富分布
      if(len(self.captial)==0):
          for  x  in  self.person:
              self.captial.append(x.captial)
      if(show):
          captial=sorted(self.captial)
          TOP20=0
          Last80=0
          n=len(self.person)
          for  i  in  range(int(0.8*n)):
              Last80+=captial[i]
          for  j  in  range(int(0.8*n),n):
              TOP20+=captial[j]
          #  print(TOP20)
          #  print(Last80)
          print("Top  20:",f'{TOP20/(TOP20+Last80)*100:.2f}',"%")
          print("Remaining  80:",f'{Last80/(TOP20+Last80)*100:.2f}',"%")
          plt.hist(captial)
          plt.yscale("log")
          plt.show()
```

```
def  getRichesAndPoverty(self):#找出所有人中最有錢和最沒錢的人
    self.CaptialDistribution()
    richest=np.argmax(self.captial)
    poorest=np.argmin(self.captial)

    return  self.person[richest],self.person[poorest]

def  printTalent(self):  #畫出所有人的talent分布
    talent=[]
    for  x  in  self.person:
        talent.append(x.talent)
    plt.title("Talent  Distribution")
    plt.hist(talent)
    plt.show()
```

Class event_map 則是生成好事地圖和壞事地圖，如果該座標有事件就標記為 1，反之標記為 0，若有多個事件在同一座標則會往上疊加數字。其他則是記錄好事和壞事的座標，和事件可以移動的方向。底下的 function 則是處理事件移動和確認事件是否超出世界邊界。和計算人物周遭九宮格內的事件數量

```
class  event_map:
    def  __init__(self,mapSize,event_num):
        self.map=np.zeros((mapSize,mapSize),dtype=np.int32)
        self.mapSize=mapSize
        self.coord=self.position(event_num)
        self.Events=np.zeros((mapSize,mapSize),dtype=np.int32)
        self.direct=self.genDirect(21)
    def  genDirect(self,n):  #生成從中心移動到21x21個位置所需要的位移量，方便計算移動後的事件座標
        matrix  =  np.arange(1,  n  *  n  +  1).reshape(n,  n)

        center_row,  center_col  =  n//2,  n//2
        moves  =  {}
        for  row  in  range(n):
                for  col  in  range(n):
                        row_diff  =  row  -  center_row
                        col_diff  =  col  -  center_col
                        moves[matrix[row,  col]]  =  [row_diff,  col_diff]
        return  moves
    def  position(self,event_num):  #生成事件的位置
        indices  =  np.random.choice(self.mapSize*self.mapSize,  event_num,  replace=False)
        points  =  np.unravel_index(indices,  (self.mapSize,self.mapSize))
        x_indices,  y_indices  =  points
        self.map[x_indices,  y_indices]  =  1
        return  list(list(a)  for  a  in  zip(x_indices,y_indices))
```

```
def checkBoundary(self,r,row): #確認移動後的座標是否超過世界邊界，如果是就需要調整
    if(r<0):
        s=-r
        return row-s ##跨過左/上邊界，跳回右/下邊
    elif(r>row-1):
        s=r-row+1
        return s-1 ##超過右/下邊界，跳回左/上邊
    else:
        return r
def move(self): #事件移動
    for event_id in range(len(self.coord)):
        dice=np.random.choice(range(1,442))
        self.map[self.coord[event_id][0]][self.coord[event_id][1]]-=1 #原本位置的好事數量減1
        self.coord[event_id][0]=self.checkBoundary(self.coord[event_id][0]+self.direct[dice][0],self.mapSize) #移動後的row值
        self.coord[event_id][1]=self.checkBoundary(self.coord[event_id][1]+self.direct[dice][1],self.mapSize) #移動後的col值
        self.map[self.coord[event_id][0]][self.coord[event_id][1]]+=1 #移動後的位置的好事加1
def printmap(self): #畫出目前事件在地圖上的分布
    print(self.coord)
    print(self.map)
def conv(self,kernelsize): #計算人物周圍九宮格內的事件數量
    kernel=np.ones((kernelsize,kernelsize),dtype=np.int32)
    padded_map = np.pad(self.map, ((1, 1), (1, 1)), mode='constant', constant_values=0)
    self.Events=convolve2d(padded_map, kernel, mode='valid')
```

由於計算周遭九宮格內的事件數量其實就是把 event_map 周遭九宮格內的數字相加起來(因為數字代表該座標發生的事件數)，其實就和 CNN 中的 convolution 的概念相同因此使用二維 convolution 實作。

Chance function 是每半年計算每個人財富的變化
從 eventmap.Events 中取出每個人所在座標周遭九宮格內有幾件好事和壞事，如果目前是計算好事，就要考量 talent 再決定要不要把財富變為 2 倍，這邊使用亂數看是否小於 talent，如果是就把財富翻倍，不是就維持原樣，壞事則直接變為原本的一半。

```
def chance(people,eventmap,mode): #計算每半年每個人經歷好事和壞事後的財富
    if(mode=="G"):    #好事
        for person in people:
            x=person.pos[0]
            y=person.pos[1]
            num=eventmap.Events[x][y]
            person.good.append(num)
            while(num):
                x=np.random.rand()
                if(person.talent>=x):
                    continue
                else:
                    person.captial*=2
                num-=1

    else:            #壞事
        for person in people:
            x=person.pos[0]
            y=person.pos[1]
            num=eventmap.Events[x][y]
            person.bad.append(-num)
            while(num):
                person.captial*=0.5

                num-=1
```

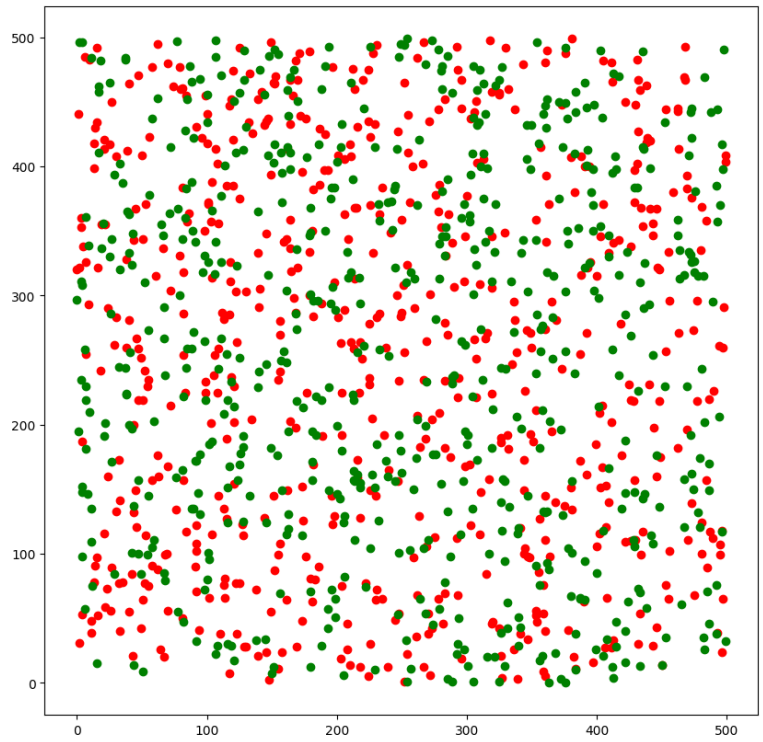
實驗結果:

模擬 10000 人，世界地圖大小為 500x500 總共 25000 個方格，並統計 40 年，每半年記錄一次財富，並 print 出財富的分布，和最有錢跟最沒錢的人的財富。

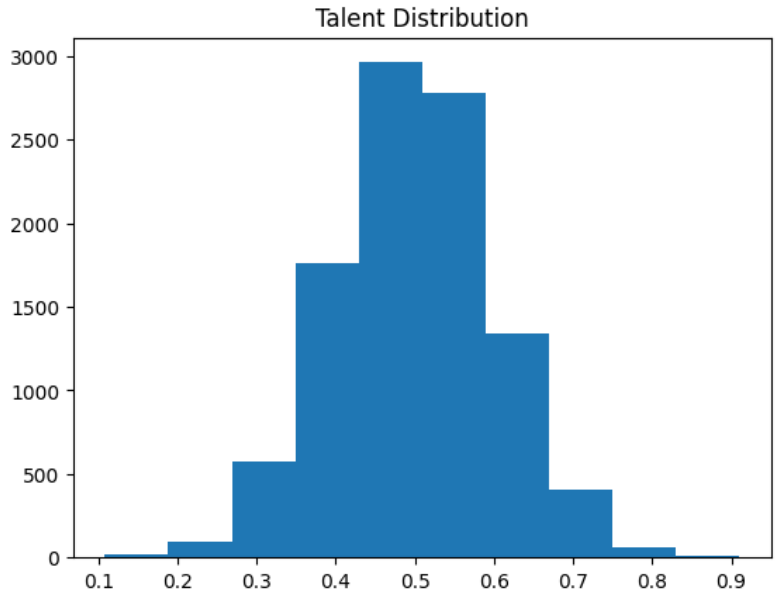
```
n=10000  #人數
event_num=n//16  #好事壞事的事件數
mapSize=500  #世界的大小500x500
year=40  #統計40年
k=year*2  #每半年統計一次財富

residents=People(n,mapSize)
good=event_map(mapSize,event_num)
bad=event_map(mapSize,event_num)
while(k):
    good.move()
    good.conv(3)
    chance(residents.person,good,mode="G")
    bad.move()
    bad.conv(3)
    chance(residents.person,bad,mode="B")
    for  x  in  residents.person:
        x.history.append(x.captial)
    k-=1
```

以下為事件分布圖(紅色為好事、綠色為壞事)

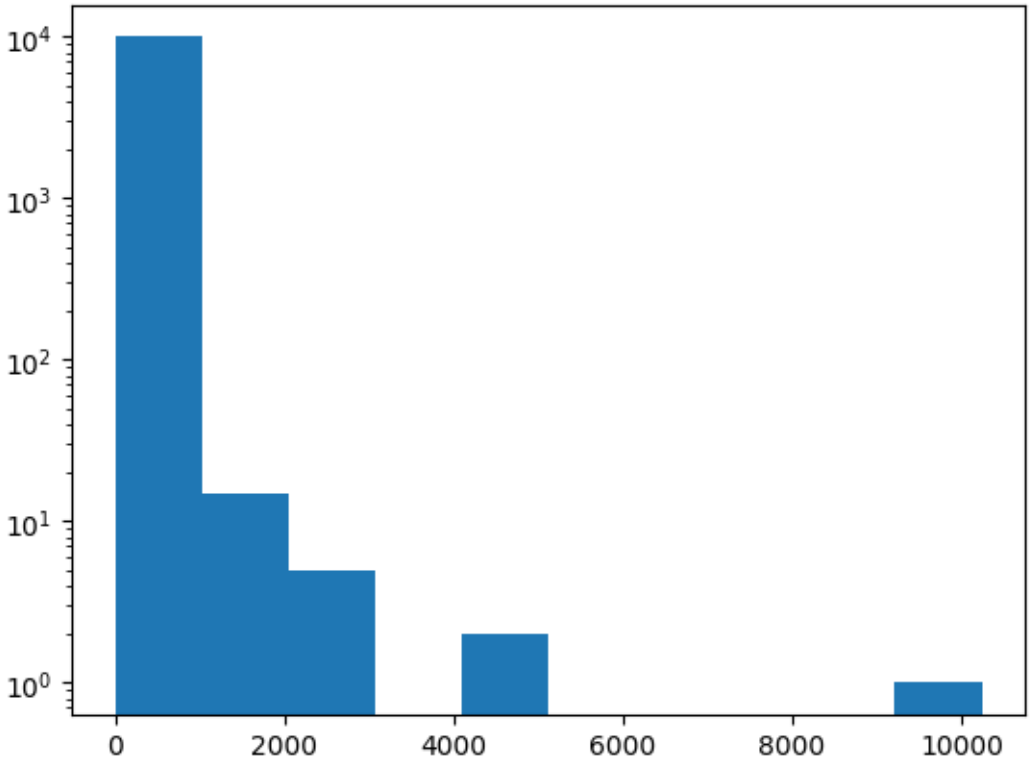


每個人的 talent 分布使用常態分布 mean 為 0.5，標準差為 0.15



結果 1:統計 40 年的財富分布，可以發現有錢的人非常少，並且呈現由前 20%的人掌握大部分財富，趨近於 8-2 法則的結果

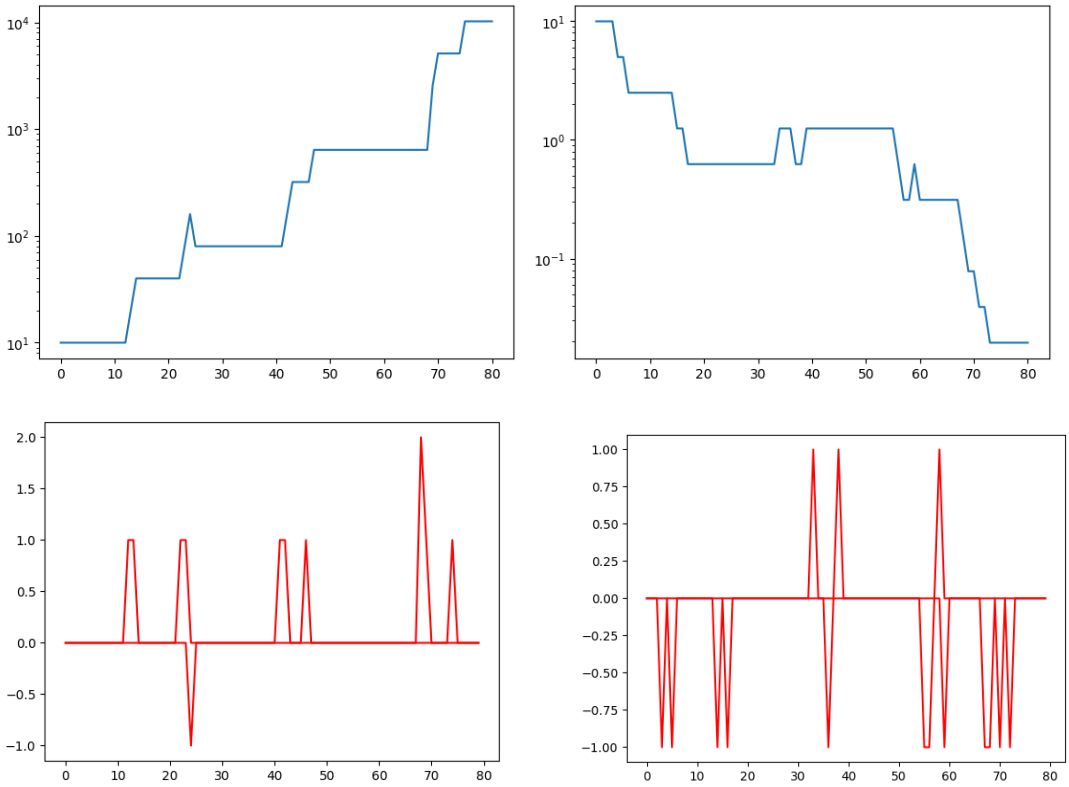
```
Top 20: 77.17 %
Remaining 80: 22.83 %
```



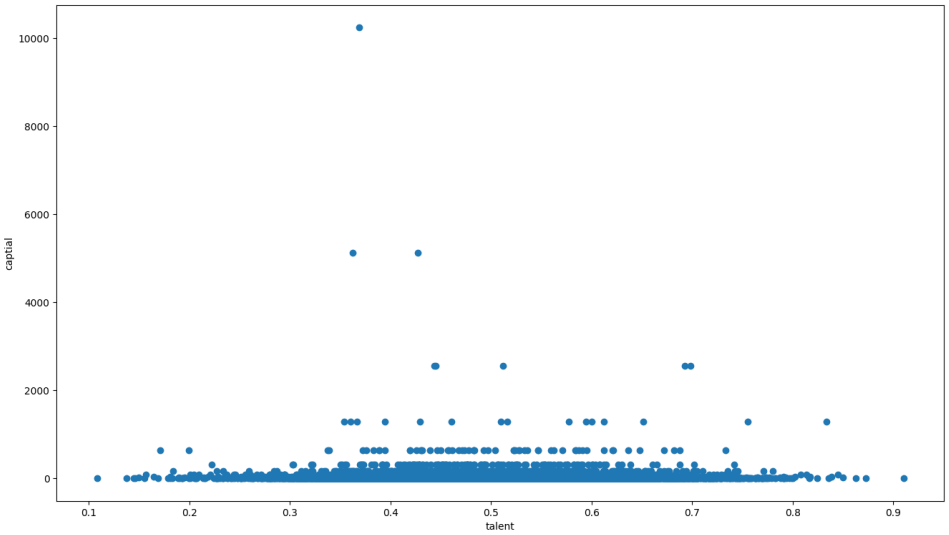
結果 2:列出最有錢和最沒錢的財富紀錄，可以發現最有錢的不一定是 talent 最高的，並且遇到的好事和壞事(運氣)確實容易影響財富的多寡

```
Richest 0.3685511933929822
10240.0
```

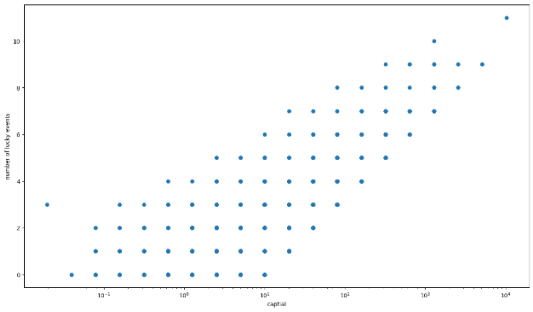
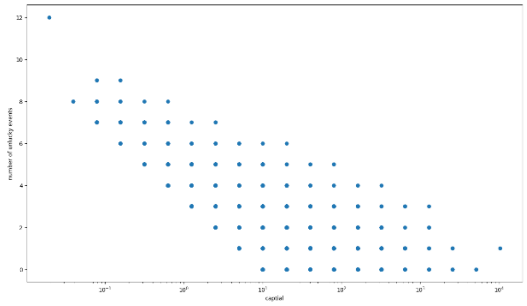
```
Poorest 0.4911893465910035
0.01953125
```



結果三:財富對應 talent 的分布，再次說明 talent 不一定是決定財富多寡的唯一原因



結果四:最終財富越多的人其遇到的好事也越多，壞事越少(運氣好)，左邊為好事數量對應財富的圖，右邊為壞事數量對應財富的圖

結果五、進行 100 次模擬並記錄每次最有錢的人和最沒錢的人的 talent
左邊為最有錢，右邊為最沒錢，有點類似常態分佈，並且最多次成為最有錢的人的通常是 talent 位於中間的人。