模擬世界：探討是什麼原因使某些人資產多?

本次模擬的假設是 1000 人，500 個事件，在一個 2 維的地圖中(由 500*500 的 2D array 表示)。每半年一次更新，共進行 40 年。

世界的規則：

人物：

人不會站在同一個位置，使用 gen_rnd_pos 自訂義 function，在 500*500 的 array 中隨機產生 1000 個位置，(row, col) pair。記到 row_col_list (p_pos_list).

Fx: Gen random row ,col

```
def gen_rnd_pos( map_width, num_point): #return list of [row, col]
    indices = np.random.choice(map_width*map_width, num_point , replace=False)
    points = np.unravel_index(indices, (map_width, map_width))
    p_row, p_col = points
    row_col_lst = np.array(list(zip(p_row, p_col)))
    return row_col_lst
```

```
## Generate 1000 unique people position
NUM_PEOPLE = 1000
MAP_SIZE = 500
p_pos_lst = gen_rnd_pos(MAP_SIZE, NUM_PEOPLE)
```

每一個人以 person object 表示，會有自己的 capital, talent, 所在位置(row, col)，capital(當下資產)，wealth_history(過去資產的紀錄)，good_events_history，bad_events_history。把所有人物 object 記到 people_list 中。

Generate People list

```
## Create people: list of person object, each person object: row, col, talent, wealth_history
class Person:
    def __init__(self, row, col, talent , capital, good_events_history, bad_events_history, wealth_history = [10] ):
        self.row = row
        self.col = col
        self.talent = talent
        self.capital = capital
        self.good_events_history = []
        self.bad_events_history = []
        self.wealth_history = wealth_history


people_list = []
```

所有人物的 talent 呈現常態分佈，平均在 0.5，標準差設在 0.166，這樣的目的是希望 99.7%的人落在[0,1]內(在三個標準差範圍內(0.002, 0998)包含 99.7%的人)，而小於 0 當作 0，大於 1 當作 1。所以 0 跟 1 的 Talent 在這世界都是可能存在的。

Talent and other attribute assign for Person

```
#Talent
talent_normal_lst = np.random.normal(0.5, 0.166, 1000)

for i in range(NUM_PEOPLE):
    if talent_normal_lst[i] < 0:
        talent_ = 0.01
    elif talent_normal_lst[i] > 1:
        talent_ = 1
    else:
        talent_ = talent_normal_lst[i]
    people_list.append(Person(p_pos_lst[i,0], p_pos_lst[i,1], talent_, 10 ,[], [], [10]))
```

事件:

- 事件的移動:

事件的移動每次以兩格為限，因此移動範圍可以想像為一個 25 宮格，中間點為現在的位置。下一步會移動到哪邊，其實是 uniform 隨機的，走每一個位置機率都相同。

Create Direction Table reference

```
direction_ref = {
    1: [-2, -2],
    2: [-2, -1],
    3: [-2, 0],
    4: [-2, 1],
    5: [-2, 2],
    6: [-1, -2],
    7: [-1, -1],
    8: [-1, 0],
    9: [-1, 1],
    10: [-1, 2],
    11: [0, -2],
    12: [0, -1],
    13: [0, 0],
    14: [0, 1],
    15: [0, 2],
    16: [1, -2],
    17: [1, -1],
    18: [1, 0],
    19: [1, 1],
    20: [1, 2],
    21: [2, -2],
    22: [2, -1],
    23: [2, 0],
    24: [2, 1],
    25: [2, 2]
}
```

Move Event function

```
# param:
def move( map, pos_list, map_size):
    for i in range(len(pos_list)):
        move_choice = rnd.randint(1,25)

        #debug
#         print(f'move_choice: {move_choice}')
        map[pos_list[i][0], pos_list[i][1]] -= 1
        pos_list[i][0] = resolve_index(MAP_SIZE, pos_list[i][0] + direction_ref[move_choice][0] )
        pos_list[i][1] = resolve_index(MAP_SIZE, pos_list[i][1] + direction_ref[move_choice][1] )
        map[pos_list[i][0], pos_list[i][1]] += 1
```

這個模擬的移動採取每次移動一些，而不是重新將事件灑點下去的原因是，認為事件發生的地點應該是以地域關係的，今天發生在台灣的好事可能是因為半導體產業，這樣的優勢不是說隔了半年就會突然跑到非洲去，因此覺得事件的移動應該是緩慢的移動的，但也知道這樣的假設，無法真正代表真實世界，畢竟這是一個簡化世界複雜度的模擬。

當事件移動到世界的邊界後，會從另一邊回來，可以想像就是把這個 2D Array 看成地圖，所代表的是一個地球，從地圖的一邊消失，其實是從另一邊出現。

Resolve index for toroidal

```
def resolve_index( width, before_index ):
    if before_index < 0:
        s = -before_index
        return width - s
    elif before_index > width-1:
        s = before_index - (width -1)
        return -1 + s
    else:
        return before_index
```

以 1 代表好事件，-1 代表壞事件。

- 事件的分佈如何產生?

Step1: 隨機生成座標位置計入(g_pos_list/ b_pos_list)，也記錄後續移動後的事件位置

Step2: 依照座標位置(row, col)放入事件紀錄圖中(good_map/ bad_map)

Generate goodmap, good_posl, bad_map, bad_posl

```
# good map of 50*50, world map is unnecessary
NUM_GOOD = 250
NUM_BAD = 250

good_map = np.full((MAP_SIZE,MAP_SIZE), 0,dtype=int) #initialize
g_pos_lst = gen_rnd_pos(MAP_SIZE, NUM_GOOD)

# place good events into good map
for i in range(len(g_pos_lst)):
    good_map[g_pos_lst[i,0], g_pos_lst[i,1]] = 1

bad_map = np.full((MAP_SIZE,MAP_SIZE), 0) #initialize
b_pos_lst = gen_rnd_pos(MAP_SIZE, NUM_BAD)

# place events into good map
for i in range(len(g_pos_lst)):
    bad_map[b_pos_lst[i,0], b_pos_lst[i,1]] = 1
```

事件可以重疊，好事，壞事都可以。這些都記在 good_map, bad_map 中，map 上的一個點值代表該位置有幾個事件。

人物遇到好事壞事，是以人的位置一個單位的半徑之內(因此是以人的位置為中心點的 9 宮格之內的事件都算入)，都會為之起反應。每個人的位置九宮格內內的好事數量，壞事數量，都會記到 good_num_map, bad_num_map 中。

Check event_num around person function: and update

```
def check_events(good_map, bad_map, people_list):
    # Padding good_map and bad_map with zeros
    padded_good_map = np.pad(good_map, 1, mode='constant')
    padded_bad_map = np.pad(bad_map, 1, mode='constant')

    good_num_map = np.zeros_like(good_map)
    bad_num_map = np.zeros_like(bad_map)

    for person in people_list:
        row, col = person.row, person.col

        # Calculate the sum within the 3x3 kernel around the person's coordinates for good_map
        kernel_sum_good = np.sum(padded_good_map[row:row + 3, col:col + 3])

        # Save the sum to the center grid's coordinate in good_num_map
        good_num_map[row, col] = kernel_sum_good

        # Calculate the sum within the 3x3 kernel around the person's coordinates for bad_map
        kernel_sum_bad = np.sum(padded_bad_map[row:row + 3, col:col + 3])

        # Save the sum to the center grid's coordinate in bad_num_map
        bad_num_map[row, col] = kernel_sum_bad

    # Call capital_update with good_num_map and bad_num_map
    capital_update(good_num_map, bad_num_map)
```

資本的變化規則(順便紀錄遇到好事或壞事):

當遇到好事件時，人物有 talent 值的機率可以把握機會將資產翻倍，因此越高 talent

值就有越高機率使 talent 翻倍。而當遇到壞事件時，想像是因為如果遇到大型意外，例如車禍，意料外的經濟危機，則無論誰都會資產減半。

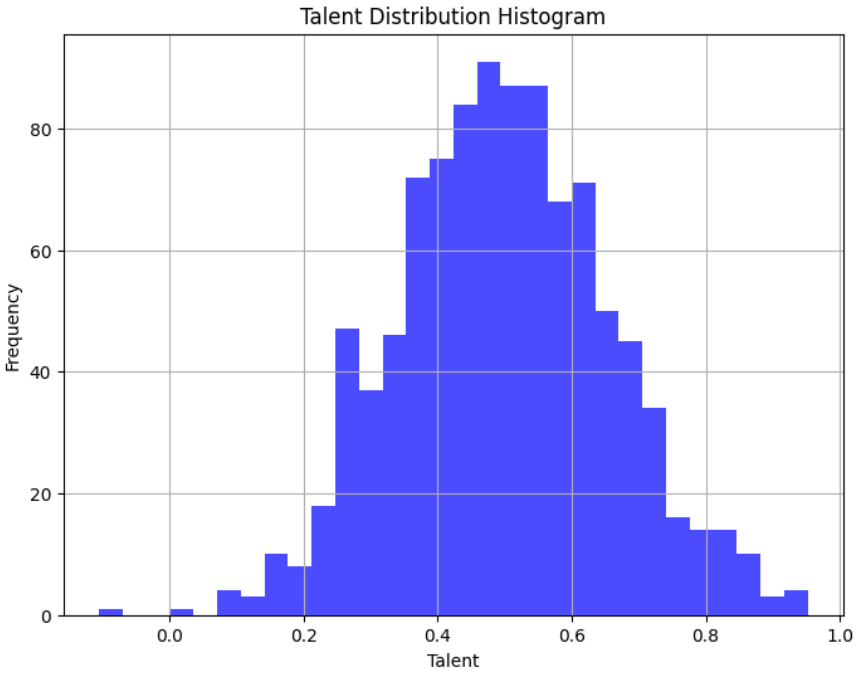
以 1 代表好事件，-1 代表壞事件。

Fx: Capital update

```
def capital_update( good_num_map, bad_num_map):
    for i in range(len(people_list)):
        good_num = good_num_map[people_list[i].row][ people_list[i].col]
        bad_num = bad_num_map[people_list[i].row] [people_list[i].col]

        people_list[i].good_events_history.append(good_num)
        people_list[i].bad_events_history.append(-1*bad_num)

        for j in range(good_num):
            u = rnd.uniform(0,1)

            #debug
            #print(f'u:{type(u)},peoplelst[i]:{people_list[i]}, talent:{type(people_list[i].talent)}')

            people_list[i].capital = people_list[i].capital * 2 if u <= people_list[i].talent else people_list[i].capital
        for k in range(bad_num):
            u = rnd.uniform(0,1)
          #try
            people_list[i].capital *= 0.5
```

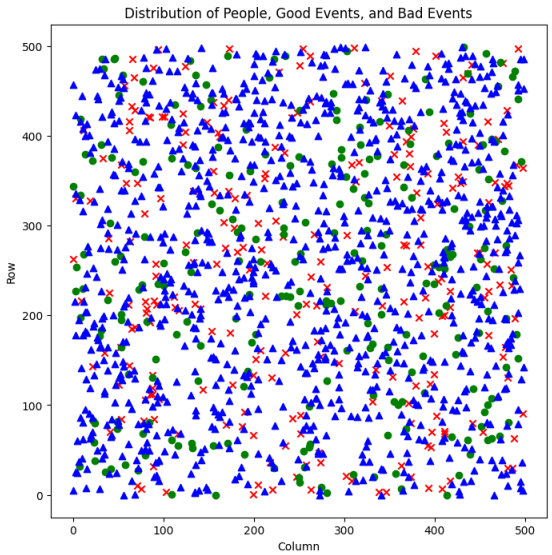
模擬結果:

Talent Distribution



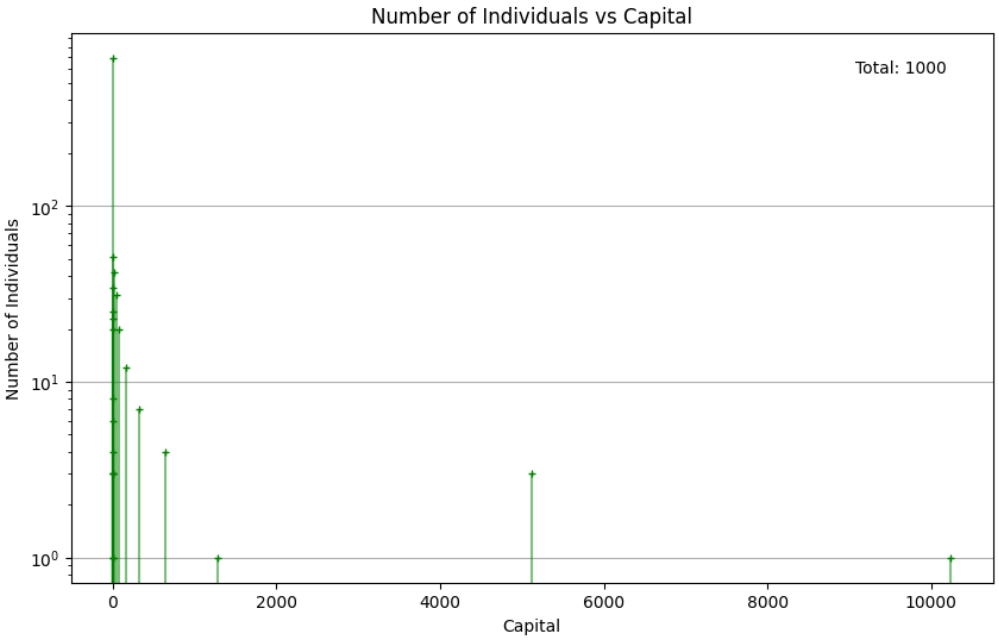
Talent 呈現常態分佈，所有資料都集中在 0 到 1 間，但 0,1 附近有高一點，因為把 0，1 以外的部分都歸到 0,1。

World_Map Visulization



Legend: 藍三角:人，綠圈:好事，紅叉:壞事

Number of individuals vs capital



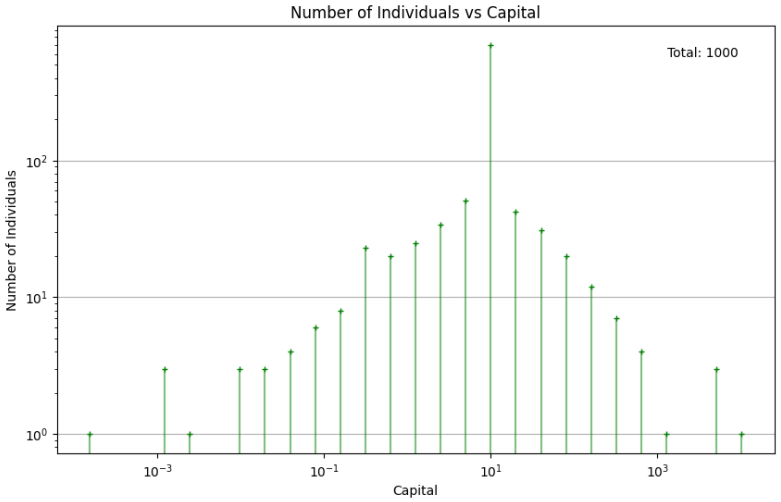
發現這樣的世界大部分都是窮人，有一個人是超級富有，大概幾十人比大多數人有錢一點，絕大部分資產接近 0。這部分的 Pareto Law 後面 log vs log 的圖中呈現。

80/20 rule?

```
Ratio of capital of top 20% to whole capital:0.8527670543779392
capital of top 20%:38070.0
capital of total population:44642.906646728516
```

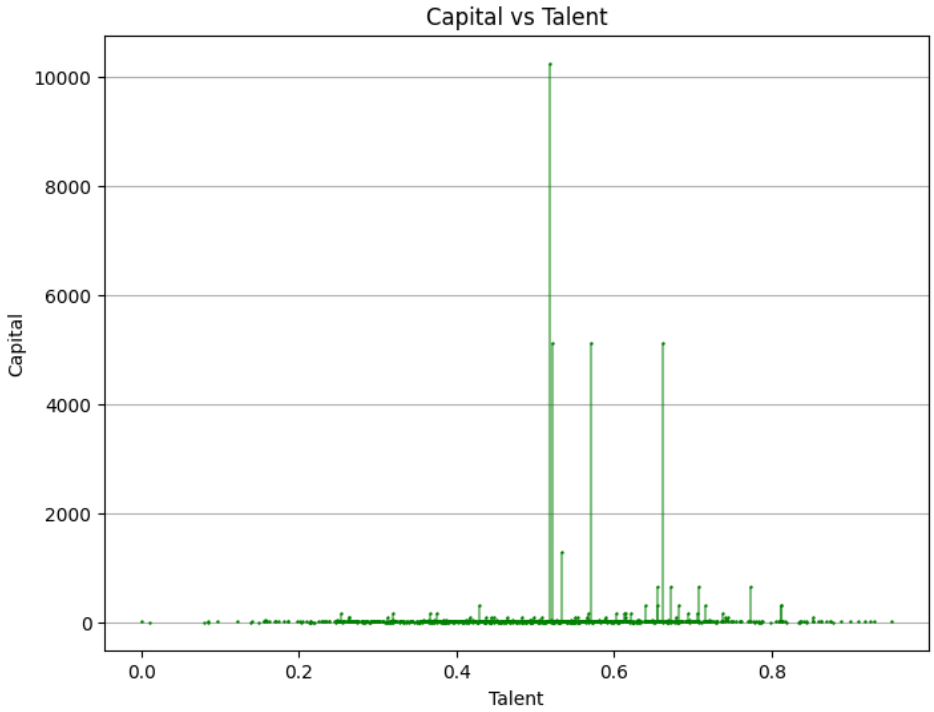
這邊可以看見前 20%的人，用有所有資產的 85%，這個範圍在 75%到 96%，表示少數人佔了大多資源。當然這只是單次的表現，並且也受到 2d array 有多大影響，在一開始的嘗試中，因為使用的地圖較小(50*50 array)，每個人可以碰到好事壞事的可能都比較大，那時的前 20%附有可能只佔了全部資產的 26%左右。

LOG Number of individuals vs LOG capital



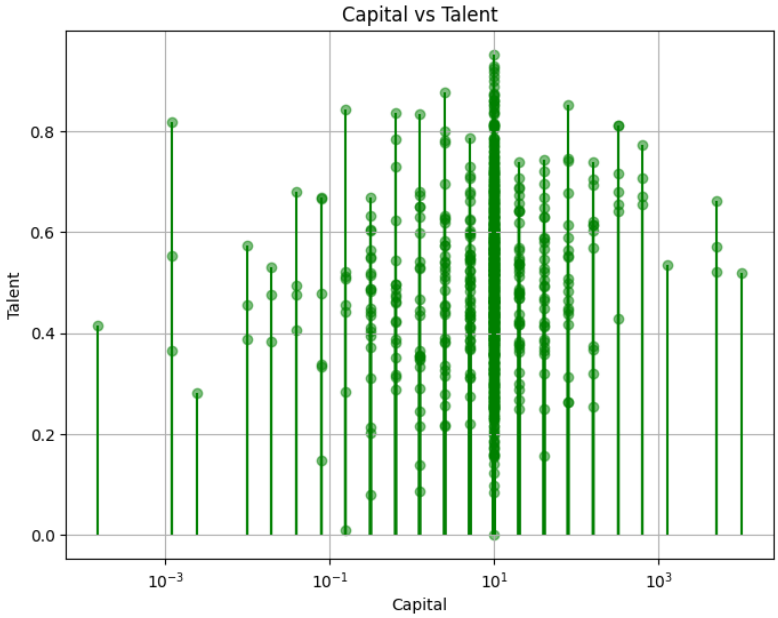
可以看到在 log, log 的圖中，capital 從10 到10^5 這段，和10^{-2}到 10 的這段都呈現線性關係，表示 number of individual vs capital 本身呈現 power law。

Capital vs talent



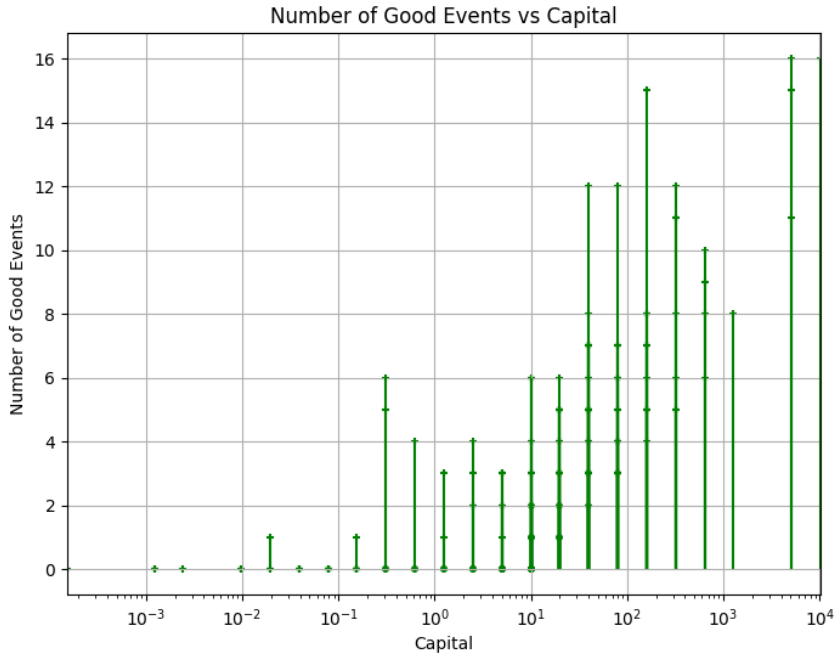
是否最有才能的人是最有錢的人？從這張圖我們可以看到，其實不是沒有這個傾向。

Talent vs capital



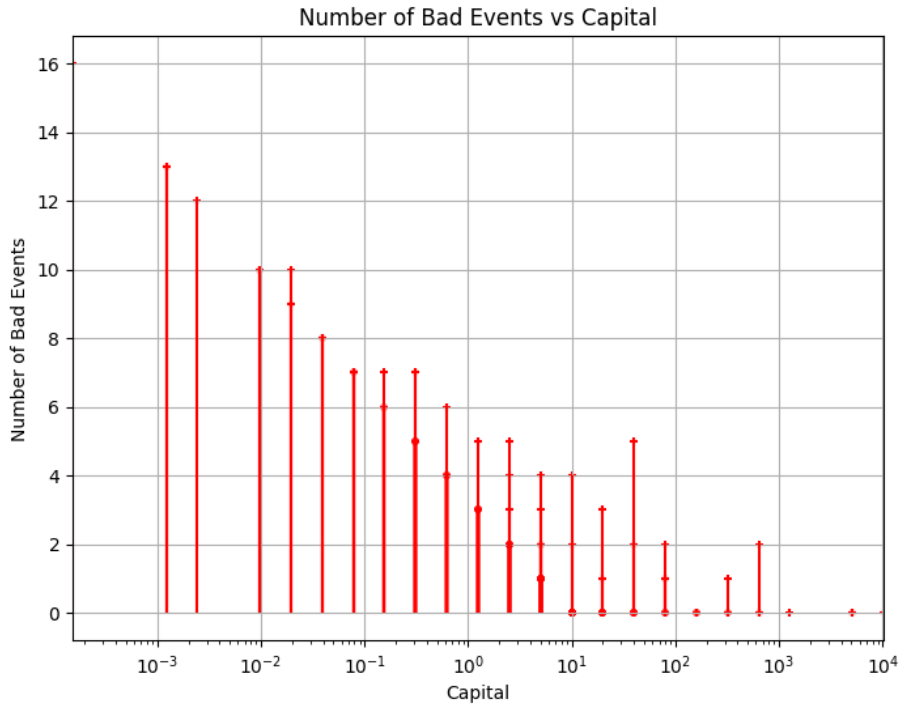
可以明顯看到大多數人不論他的才能，這輩子的資產維持在一開始的樣子。而且是否富有看起來跟才能值沒太大關係。我們看到最富有的人，他的才能值業就是 0.5。並沒有特別突出。也看到本次模擬當中最有才能的那個人，他的資產甚至還是停留在一開始的樣子，就是十。

Number of lucky events vs capital



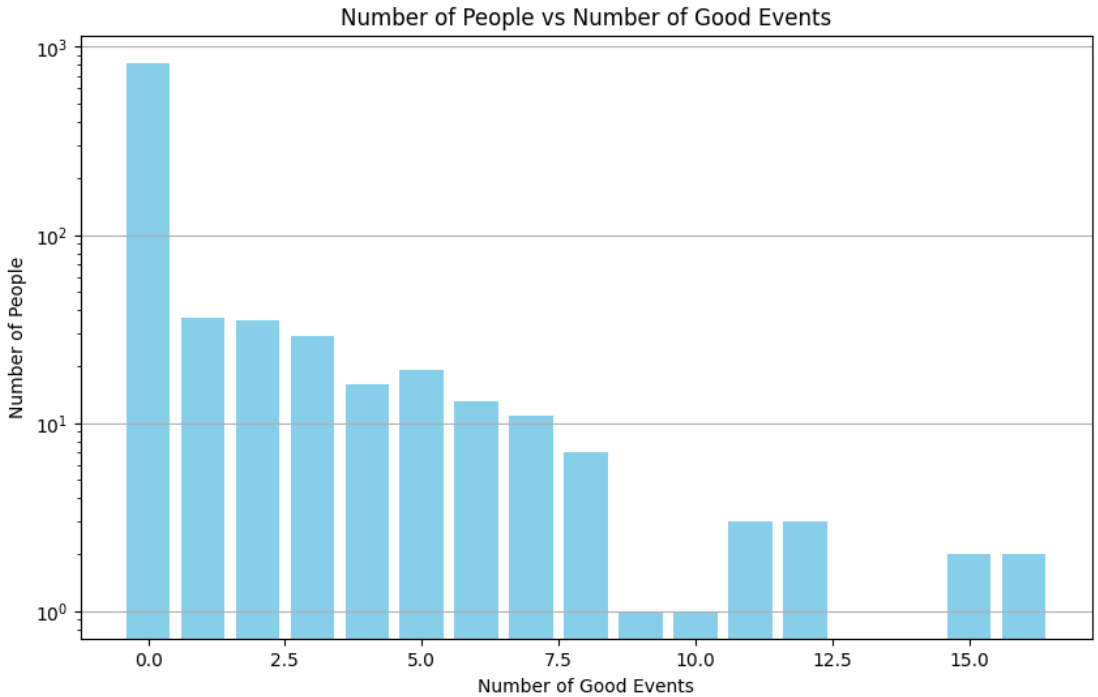
可以看到資產 1 塊錢到資產 10 萬越來越富有的區段，他們所遭遇到的好事的數量也是越來越多，因此我們可以推測遇到好事越多，越容易富有這件事情。甚至可以推測。或許真的讓這些人富有的原因，有一大部分就是運氣。

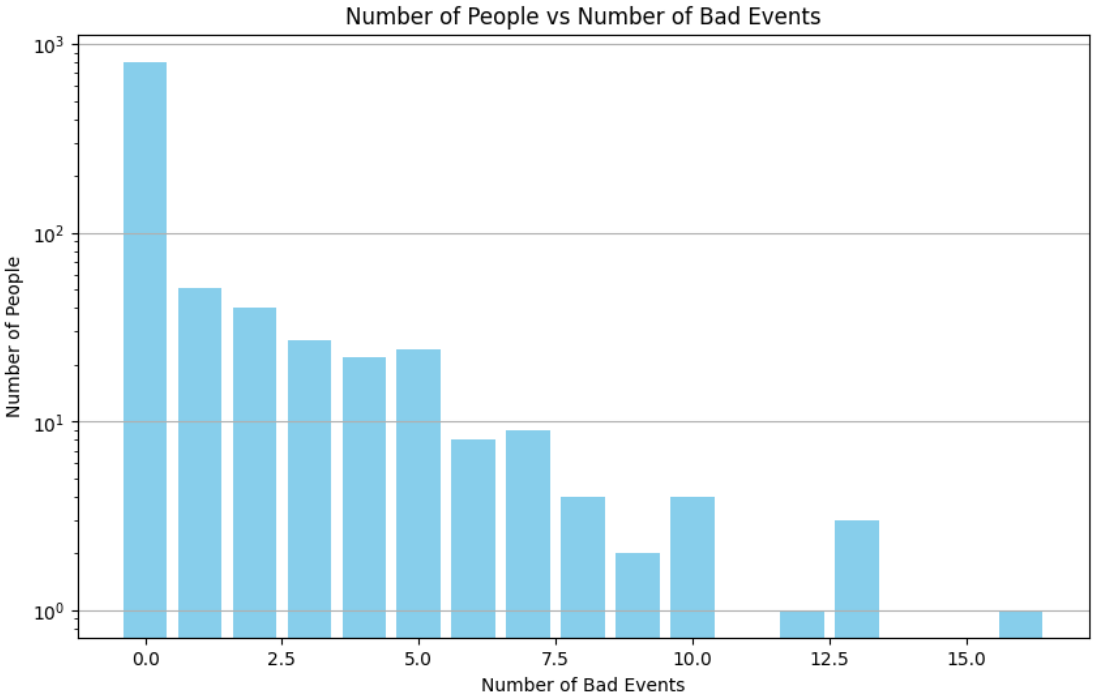
Number of Bad events vs capital



可以發現，資產值越小的人，他們這一輩子也遇到了壞事也更多。我們可以大致推測，遇到更多壞事的人，更傾向於變得窮。

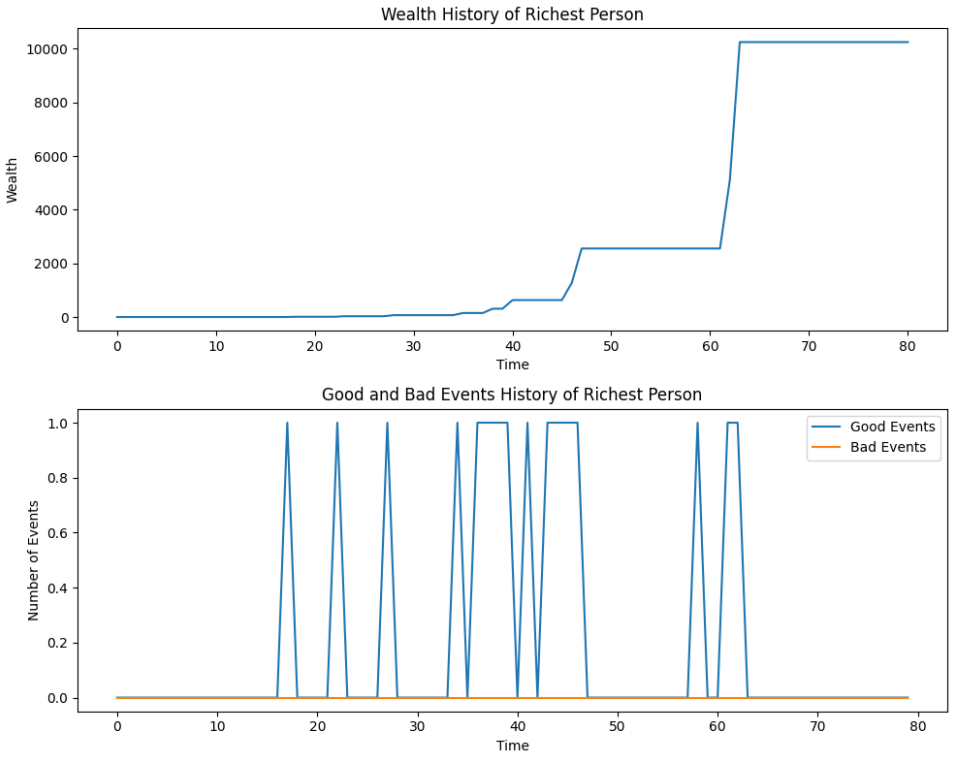
Number of agents vs good_events and bad_events.





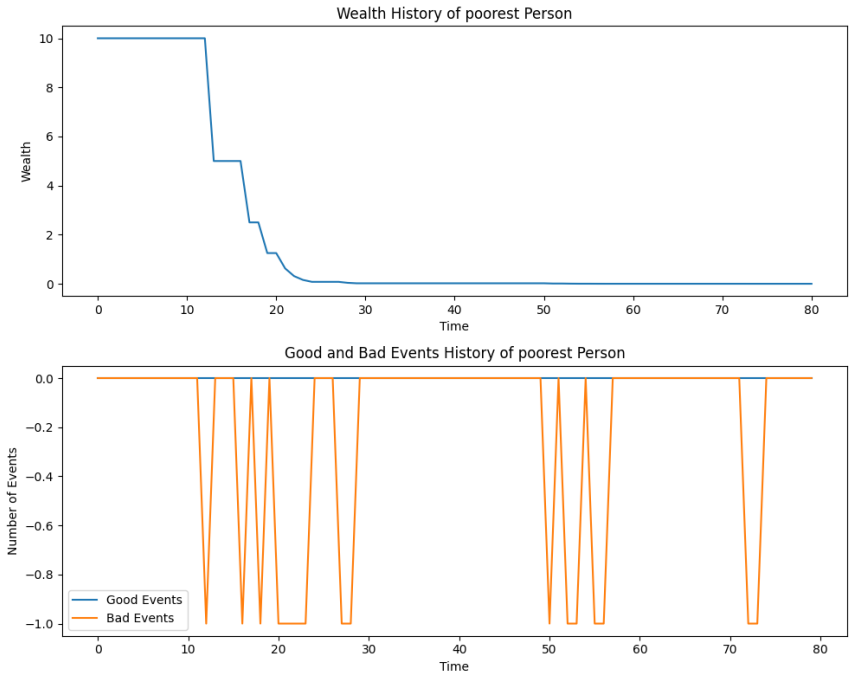
大致上遇到好事跟壞事的人數分布差不多，並且我們可以注意到，在我們的模擬當中。沒有遇到任何好事跟壞事的人其實佔了大多數，這些沒有經歷過大起大落的人接近了 1000 人，而我們模擬人數也就 1000 人。遇到越多事件的人越少。

Most successful lifetime capital vs time



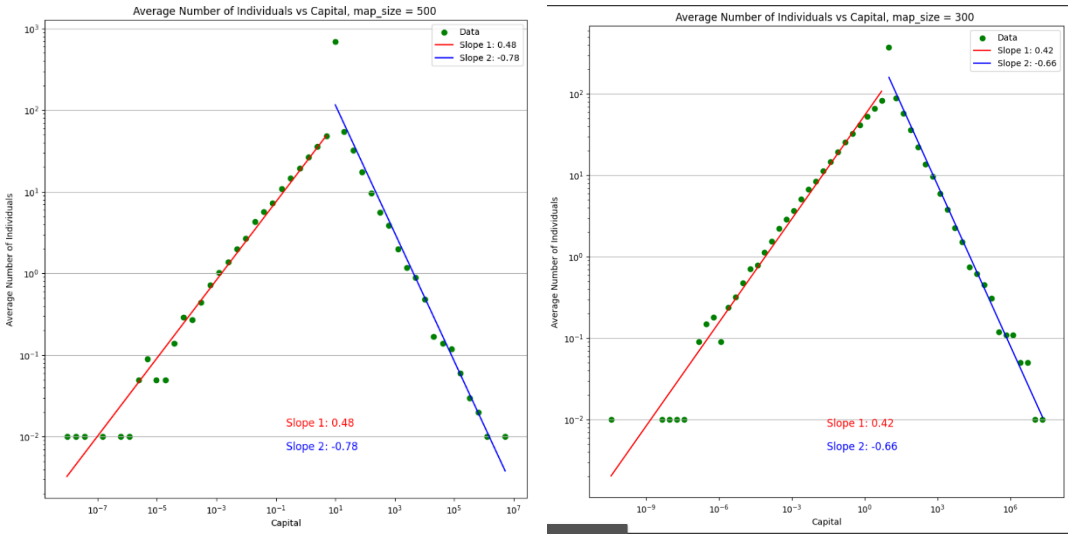
這是最富有的人可以看到。前 25 次迭代當中，遇到了一些好事，但他沒有抓住，但儘管如此，因為他這輩子也沒有遇到任何壞事，所以在第 20 年之後，當他在遇到一些好事，他的資產不斷的翻倍，有抓住機會。

Less successful lifetime capital vs time



最慘的這個人，在大概工作的第五年開始就遇到一些壞事資產歸零，到第 25 年左右又遇到一系列壞事，一值處在 0。這是一個極端值，顯示因為他這輩子沒遇到什麼好事，因此他的資產值不斷的下降。

多次模擬取平均



Run 100 次以後，取平均，觀察不同資產擁有者的數量(log vs log)，我們應

該要觀察的點是由 capital 等於 10 到 max capital 的這一段，首先在一個 log,log 圖中的斜率為直線可以說明有 x 與 y 有 power law 的關係，但如果要符合 pareto distribution，斜率要在-1 到-3 之間，而這個模擬的斜率在-0.78，沒有嚴格符合，但從前述取前 20%富有者佔有總體人口資產的 70%到 96%還是可以看出少數人掌握大多數資源的樣貌。觀察在 map_size 的情況下，更沒有符合 pareto distriution，map_size = 500 時較符合。模擬中，因為地圖的大小就會去影響有沒有符合 pareto law 這件事本身其實就表示，其實地圖的大小在本次模擬中沒有什麼標準答案，也沒有真的能夠表示真實世界的什麼面積比例關係，因此，只能說依照我們簡化過後的資產變化規則，看到的一些結果。

另外，可能的原因也是，我們對於事件的移動這件事情的模擬，採用的是有點像布朗運度的方式在移動，而且還限制其能夠移動的範圍，這本身與世界真實的樣貌，就有些出入，我們也無從得知真實事件是如何移動的。