hw1

1. → data: X & parameter of interest: θ

→ decision procedure $\delta(x)$ & loss function $l(\theta, \delta(x))$

a. → # german tank | k = captured tank.

→ serial number = $\#^{th}$ produced.

→ decision: $m + (m-k)/k$

⇒ data X: the serial numbers of captured tank

⇒ parameter of interest: the total number of German tank that had been produce

⇒ decision procedure $f(x)$ = total produced $= m + \frac{m-k}{k}$ ← $m = \max(\text{serial})$.

⇒ loss function: actual total produced tank − predicted total from serial number

b. → adaptive (not neutral) mutation

→ adaptive = $P(N \mid \text{nature mutation rate}) <= \text{threshold}(\eta)$

→ cost → C_{01} (adaptive as neutral)
→ C_{10} (neutrue as adaptive)

⇒ data: # same mutation ocured independently in N different species.

⇒ loss function → C_{01} (predicted adaptive | neutrue)
→ C_{10} (predicted neutrue | adaptive)

⇒ interested parameter: whether the mutation is adaptive

$\Rightarrow \delta(x) \begin{cases} \text{if } P(N \mid rate^*) < y \rightarrow \text{adaptive mutation.} \\ \text{if } P(N \mid rate^*) >= y \rightarrow \text{neutral mutation.} \end{cases}$

c.

$$\Rightarrow \min_{\delta} E_{(x,\theta)\sim P}[l(\theta, \delta(x))] = \min_{\delta} \mathbb{E}_{(x,\theta)\sim P}[(\theta - \delta(x))^2]$$

$$= \min_{\delta} \mathbb{E}_{x\sim P(x)}\left[\mathbb{E}_{\theta\sim P(\theta|x)}[(\theta - \delta(x))^2]\right]$$

$\Rightarrow$ let $\delta(x) = a$

$$\Rightarrow f'(a) = \mathbb{E}_{\theta\sim P(\theta|x)}[2*(\theta - a)] = 0$$

$$= \mathbb{E}_{\theta\sim P(\theta|x)}(2\theta) - \mathbb{E}_{\theta\sim P(\theta|x)}(2a) = 0$$

$$\Rightarrow \mathbb{E}_{\theta\sim P(\theta|x)}[\theta] = a$$

$$\Rightarrow \delta(x) = \mathbb{E}_{\theta\sim P(\theta|x)}[\theta]$$

$\Rightarrow$ thus, decision function is $\delta(x) = \mathbb{E}_{\theta\sim P(\theta|x)}[\theta]$.

d.

2. → X = uniform dist under null

→ $\mathbb{P}(X \mid H_0) \begin{cases} 1 & 0 \le x \le 1 \\ 0 & \text{otherwise} \end{cases}$

→ $\mathbb{P}(X \mid H_1) \begin{cases} 2x & 0 \le x \le 1 \\ 0 & \text{otherwise.} \end{cases}$

→ prior prob X = uniform $(0, 1)$ dist.

→ $C_{00} = C_{11} = 0$ & $C_{10}, C_{01} \in [0, \infty]$.

→ C_{ij} → predicted i → reality j.

a. → decision rule that minimize error prob.

⇒ minimize probability of error = → max TPR → minimize TPR

⇒ let's set the significant level to be α.

⇒ fits the neyman-pearson lemma

⇒ likelyhood ratio $= \dfrac{f_{\theta_0}(x)}{f_{\theta_1}(x)}$

⇒ null hypothesis = uniform dist → $f_{\theta_0}(x) = x$

⇒ alternative hypothesis = $\int 2x\,dx = 2 * \frac{x^2}{2} = x^2$

⇒ likelyhood ratio $= \dfrac{x}{2x^2} = \dfrac{1}{2x}$

⇒ $P(\frac{1}{2x} \le \eta \mid H_0) = \alpha$

⇒ $P(\frac{1}{x} \le 2\eta \mid H_0) = \alpha$ → $P(x \ge \frac{1}{2\eta} \mid H_0) = \alpha$

⇒ $\int_{\cdot}^{1} x\,dx = \alpha$

$1/2y$

$$\left[\frac{x^2}{2}\right]_{\frac{1}{2y}}^{1} = \alpha$$

$$\left[\frac{1}{2} - \left(\frac{1}{4y^2}/2\right)\right] = \alpha$$

$$\left[\frac{1}{2} - \frac{1}{8y^2}\right] = \alpha$$

$$\frac{1}{8y^2} = \frac{1}{2} - \alpha$$

$$\frac{1}{y^2} = 4 - 8\alpha$$

$$y = \frac{1}{\sqrt{4-8\alpha}}$$

$\Rightarrow$ y have to be $>0 \rightarrow 4-8\alpha > 0$.

$\Rightarrow 4 > 8\alpha \rightarrow \alpha < \frac{1}{2}$

$\Rightarrow$ for minimizing the probability error, set $\lim_{\alpha \to 0}$

$\Rightarrow$ at $\lim_{\alpha} = 0 \rightarrow y = \frac{1}{\sqrt{4-0}} = \frac{1}{2}$

$\Rightarrow$ thus, $y = \frac{1}{2}$

$\Rightarrow$ thus, the decision rule is

$$\Rightarrow \delta(x) = \begin{cases} \text{reject null} = \dfrac{f_{\theta_0}(x)}{f_{\theta_1}(x)} \leq \dfrac{1}{2} \\ \text{accept null} = \dfrac{f_{\theta_0}(x)}{f_{\theta_1}(x)} > \dfrac{1}{2}. \end{cases}$$

b. $\rightarrow P_D = \mathbb{P}(\hat{H} = H_1 \mid H = H_1)$

$\rightarrow P_F = \mathbb{P}(\hat{H} = H_1 \mid H = H_0)$.

$\rightarrow P_D$ as a func of P_F for likelihood ratio test (LRT).

$$\Rightarrow P_D = \mathbb{P}(\hat{H}=H_1 \mid H=H_1)$$

$$= TPR = \int \delta(x)\,\mathbb{P}(x\mid\theta_1)\,dx$$

$$= \mathbb{E}_{x\sim\mathbb{P}(x\mid\theta_1)}[\delta(x)]$$

$$\Rightarrow P_F = \mathbb{P}(\hat{H}=H_1 \mid H=H_0) = FPR$$

$$= \int \delta(x)\,\mathbb{P}(x\mid\theta_0)\,dx$$

$$= \mathbb{E}_{x\sim\mathbb{P}(x\mid\theta_0)}[\delta(x)]$$

$$\Rightarrow P'_F = \delta(x)\,\mathbb{P}(x\mid\theta_0)$$

$$\Rightarrow P'_D = \delta(x)\,\mathbb{P}(x\mid\theta_1).$$

$$\Rightarrow L(x) = \frac{\mathbb{P}(x\mid\theta_1)}{\mathbb{P}(x\mid\theta_0)} = \frac{P'_D}{P'_F}$$

$$\Rightarrow L(x) = \frac{\mathbb{P}(x\mid H_1)}{\mathbb{P}(x\mid H_0)} = \frac{P'_D}{P'_F} = \frac{2x}{1} = 2x$$

$$\Rightarrow P'_F = \delta(x)\,\mathbb{P}(x\mid\theta_0) \;\&\; P'_D = \delta(x)\,\mathbb{P}(x\mid\theta_1)$$

$$\Rightarrow P'_D = P'_F * \frac{P'_D}{P'_F} = P'_F * 2x$$

$$\Rightarrow P'_D = \delta(x)\,\mathbb{P}(x\mid\theta_0) * 2x$$

$$\Rightarrow P_D = \int \delta(x)\,\mathbb{P}(x\mid\theta_0) * 2x\,dx$$

$$= 2\int \delta(x)\,\mathbb{P}(x\mid\theta_0)\,x\,dx$$

$$= 2 * \left[x * P'_F - P_F\right]$$

C.

$\Rightarrow$ the statement is false.

3. $\rightarrow N = 1000$ hypothesis & $H_0 \in \{1 \ldots N\}$ of all null

$\rightarrow \pi_0$ = prop true null = $|H_0| / N$. & P_i = corresponding P-value.

$\rightarrow P_i$ = alter = 0.01

$\rightarrow P_i \in$ null = uniform [0,1] dist

$\rightarrow \alpha = 0.05$.

a. $\rightarrow$ FDR in terms of π_0, N & α

$\rightarrow$ # $P_i \leq \alpha$ related to $N\pi_0$ trials

$\rightarrow$ each trail success with probability $\mathbb{P}(P_i \leq \alpha) = \alpha$

$\rightarrow$ special dist. & reject $H_i \in P_i \leq \alpha$.

$\Rightarrow$ FDR = # false positive / # all rejected (1) hypothesis.

$\Rightarrow$ # alternative hypothesis = $N - N\pi_0$

$\Rightarrow$ since $\alpha = 0.05$ & P_i for H_i & $H_0 = 0.01$

$\Rightarrow$ our decision is going to reject all alternative (reality hypothesis).

$\Rightarrow$ for each null trail $\mathbb{P}(P_i < \alpha) = \alpha$ & α is small

$\Rightarrow$ the probability that there are k rejected null hypothesis is

a binomial distribution

$\Rightarrow \mathbb{P}(\text{\# false positive} = k) = \binom{N\pi_0}{k} \alpha^k (1-\alpha)^{N\pi_0 - k}$

$\Rightarrow$ let $k =$ # false positive

$\Rightarrow FDR = \frac{k}{k + \text{\# true negative}} = \frac{k}{k + N - N\lambda_0}$

$\Rightarrow FDR = E\left(\frac{FP}{FP + TP}\right)$ & TP is a constant

$\Rightarrow$ since FP have a poisson distribution

$$\Rightarrow E\left(\frac{k}{k+c}\right) = E\left(k * \left(\frac{1}{k+c}\right)\right)$$

$$= \sum_{k=0}^{N\lambda_0} \binom{N\lambda_0}{k} \alpha^k (1-\alpha)^{N\lambda_0 - k} * k * \frac{1}{k + N - N\lambda_0}$$

$$= \frac{N\lambda_0 \alpha}{N\lambda_0 \alpha - N\lambda_0}$$

$\Rightarrow$ thus, the $FDR = \frac{N\lambda_0 \alpha}{N\lambda_0 \alpha - N\lambda_0}$

b.

$$\Rightarrow E[\text{sensitivity}] = E\left[\frac{\text{\# true discovery}}{N(1-\lambda_0)}\right]$$

$$= \frac{1}{N(1-\lambda_0)} * E[\text{\# TP}]$$

$$= 1$$

$\Rightarrow$ thus, the expected sensitivity of this decision rule is 1.

c.

$\Rightarrow$ for bonferroni correction $y = \frac{\alpha}{N} = \frac{0.05}{1000}$

$\Rightarrow$ reject H_i if $P_i \leq \frac{0.05}{1000} = 0.00005$

$\Rightarrow FDR =$ # false positive / # all rejected (1) hypothesis.

$$= \frac{FP}{FP+TP}$$

$\Rightarrow$ since $0.00005 > 0.00001 \rightarrow TP = 0$

$$\Rightarrow P(FP=0) = P(\text{all } P_i > \frac{\alpha}{N})$$
$$= (1-\frac{\alpha}{N})^{N\lambda_0}$$

$\Rightarrow$ if $FP \neq 0 \Rightarrow FDR = 1$

$$\Rightarrow P(FDR=1) = P(FP \neq 0) = 1 - (1-\frac{\alpha}{N})^{N\lambda_0}$$

$$\Rightarrow FDR = 0 * (1-\frac{\alpha}{N})^{N\lambda_0} + 1 * [1-(1-\frac{\alpha}{N})^{N\lambda_0}]$$
$$= 1 - (1-\frac{\alpha}{N})^{N\lambda_0}$$

$$\Rightarrow E(\text{sensitivity}) = E(\frac{TP}{TP+FN}), \leftarrow TP = 0$$
$$= 0$$

$\Rightarrow$ thus, the expected sensitivity is 0.

d.

$\Rightarrow$ cutoff $= 0.00005$

$\Rightarrow$ in order to discover all (small enough P_i to get rejected).

$P_i \in H_{alternative} \leq 0.00005$

$\Rightarrow$ thus, I need to decrease the P_i for alternative hypothesis, in order for the procedure to discover them all.

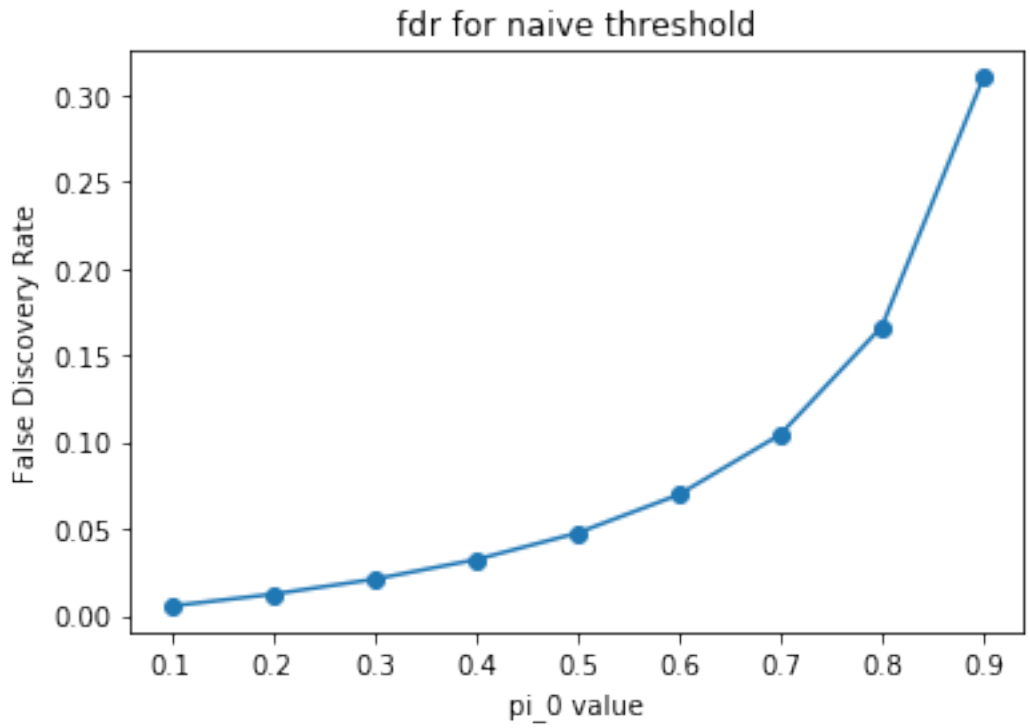
e.

⇒ data type error

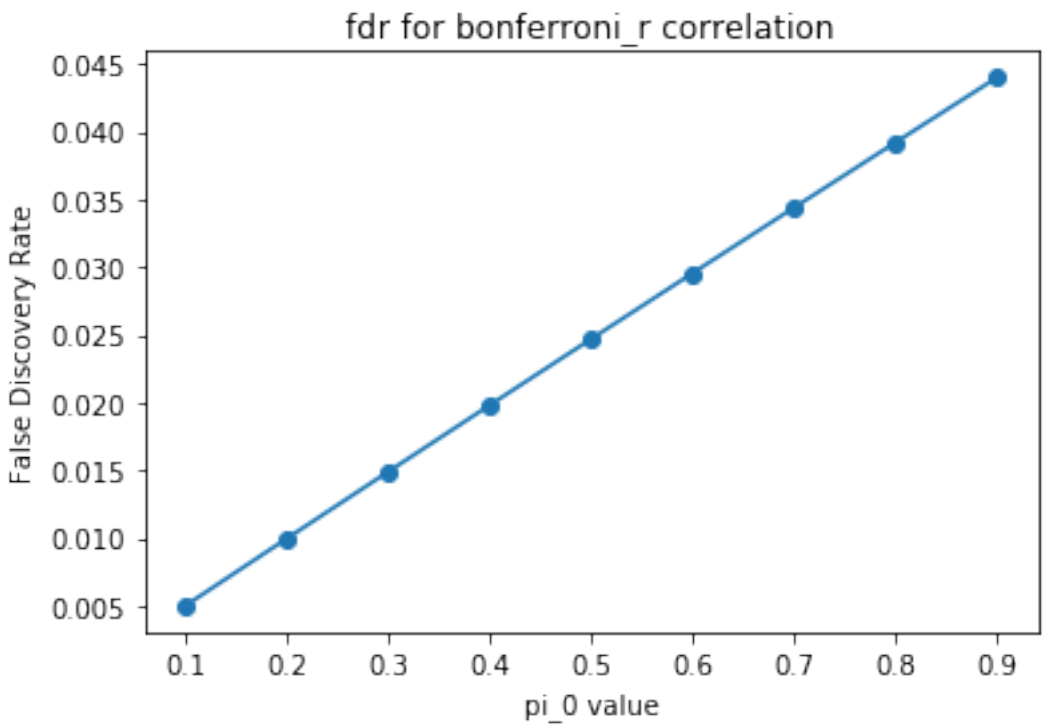
Untitled

February 11, 2020

[5]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd


def fd_r(portion_list):
    N= 1000
    alpha = 0.05
    output = []
    for pi in portion_list:
        new_fdr = (N*pi*alpha)/(N*pi*alpha+(N-N*pi))
        #output = output + [new_fdr]
        output = np.append(output,new_fdr)
    return output
input_list = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
fd_r(input_list)

plt.plot(input_list,fd_r(input_list),'-o')
plt.xlabel('pi_0 value')
plt.ylabel('False Discovery Rate')
plt.title('fdr for naive threshold')
plt.show()
```



[8]:
```
def bonferroni_r(list_2):
    N= 1000
    alpha = 0.05
    output = []
    cutoff = alpha / N
    for pi in list_2:
        new_fdr = 1 - ((1 - cutoff)**(N*pi))
        output = np.append(output,new_fdr)
    return output
#bonferroni_r(input_list)

plt.plot(input_list,bonferroni_r(input_list),'-o')
plt.xlabel('pi_0 value')
plt.ylabel('False Discovery Rate')
plt.title('fdr for bonferroni_r correlation')
plt.show()
```



[36]:
```
def population_gen(list_3):
    N = 1000
    pi_fdr_holder = []
    pi_sensitivity_holder = []
    for pi in list_3:
        fdr_holder = []
        sensitivity_holder = []
        for i in np.arange(100):
            alternative_p = [0.01] * int((N - N*pi))
            null_p = np.random.uniform(0,1,int(N * pi))
            alternative_p.extend(null_p)
            total_state = [1] * int((N - N * pi))
            total_state.extend([0] * int((N * pi)))
            alg_result = benjamini_alg(alternative_p, 0.05)
            TP = sum((alg_result == 1) & (total_state== 1))
            TN = sum((alg_result == 0) & (total_state== 0))
            FP = sum((alg_result == 1) & (total_state== 0))
            FN = sum((alg_result == 0) & (total_state== 1))
            sensitivity = TP / (TP + FN)
            FDR = FP / (FP + TP)
            fdr_holder = fdr_holder + FDR
            sensitivity_holder = sensitivity_holder + sensitivity
        pi_fdr_holder = np.append(pi_fdr_holder, np.mean(fdr_holder))
```

```
        pi_sensitivity_holder = np.append(pi_sensitivity_holder,np.
 ↪mean(sensitivity_holder))
    return [pi_fdr_holder,pi_sensitivity_holder]


def benjamini_alg(p_values, alpha):
    n = len(p_values)
    modify = p_values.copy()
    modify.sort()
    track = 0
    for k in np.arange(1,n+1):
        calculate = k * (alpha / n)
        if modify[k - 1] <= calculate:
            track = track + 1
        else:
            break
    cut_off = modify[track-1]
    decisions = np.array(np.asarray(p_values) <= cut_off)
    #print(track)

    return decisions


a = population_gen(input_list)
a
```

```
/srv/conda/envs/data102/lib/python3.7/site-packages/ipykernel_launcher.py:19:
RuntimeWarning: invalid value encountered in long_scalars
/srv/conda/envs/data102/lib/python3.7/site-packages/ipykernel_launcher.py:20:
RuntimeWarning: invalid value encountered in long_scalars
```

[36]:
```
[array([nan, nan, nan, nan, nan, nan, nan, nan, nan]),
 array([nan, nan, nan, nan, nan, nan, nan, nan, nan])]
```

[30]:
```
#null_p = np.random.uniform(0,1,int(1000 * 60)).tolist()
#alternative_p = [0.01] * int((1000 - 1000*0.2))
#alternative_p.extend(null_p)
#alternative_p
```

[30]:
```
[0.01,
 0.01,
 0.01,
 0.01,
 0.01,
 0.01,
 0.01,
```

```
0.01,
0.01,
0.01,
```

```
0.01,
0.01,
0.01,
```

```
0.01,
0.01,
0.01,
```

```
0.01,
0.01,
0.01,
```

```
0.01,
0.01,
0.01,
```

```
0.01,
0.01,
0.01,
```

```
0.01,
0.01,
0.01,
```

```
0.01,
0.01,
0.01,
```

```
0.01,
0.01,
0.01,
```

```
0.01,
0.01,
0.01,
```

```
0.01,
0.01,
0.01,
```

```
0.01,
0.01,
0.01,
```

```
0.01,
0.01,
0.01,
```

```
0.01,
0.01,
0.01,
```

```
0.01,
0.01,
0.01,
```

```
0.01,
0.01,
0.01,
```

```
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.01,
0.5998480362520013,
0.2282865913820301,
0.5872981448629974,
0.6963007339591714,
0.14576680705705702,
0.7710234662850303,
```

0.3822388866035227,
0.14128563935949323,
0.7973786843402042,
0.5199368660272443,
0.20801238320924897,
0.35602664303503595,
0.6400496425237561,
0.7758695274158817,
0.10175878988036946,
0.1689495555776669,
0.700328867834934,
0.8689666875560673,
0.07456814019767855,
0.13595567855650614,
0.5379644350838627,
0.3247452908271742,
0.14554624178667075,
0.4512038812671154,
0.2868683306390769,
0.09611921376771937,
0.7423044973216867,
0.08191414648529871,
0.9436218176131822,
0.9615395827281696,
0.6186684110447946,
0.8956230074230126,
0.350532300421213,
0.9456008225823337,
0.4089660604017452,
0.6783407993622826,
0.08596698218414722,
0.9321591482265219,
0.016364708920836857,
0.905006780055915,
0.8904069454633361,
0.14900190889614529,
0.518295340590097,
0.9200563702640376,
0.5417368439114593,
0.2549152858723397,
0.19465754911541766,
0.5092188431009822,
0.12974317908727206,
0.6136523669038292,
0.9972155662705967,
0.9992975428767685,
0.54116490209271,

0.8985305255057943,
0.4170806511510766,
0.0008566834734807793,
0.8296432559686339,
0.8629493708898881,
0.120872567194801,
0.6325109223409175,
0.7390931814520075,
0.46952731087921595,
0.2798677877726392,
0.03462321696418036,
0.4387888051744463,
0.05870151393369616,
0.5924525802586795,
0.47630645488424106,
0.663420943650373,
0.4278541141472638,
0.7889426716624883,
0.08683978712419138,
0.4509640502333998,
0.9944305489475118,
0.9891070951817081,
0.7891735176804513,
0.9639952272562684,
0.30666808814279767,
0.6972499764516198,
0.1127525503650757,
0.9915923971333126,
0.18053290041982262,
0.7378038060981192,
0.7968631758758884,
0.4736479314724614,
0.09443546121764546,
0.7094908236276587,
0.12699487610903026,
0.19253491640245746,
0.9949571173000221,
0.28922941475993214,
0.9705096646160576,
0.8006324846049464,
0.05107209191115025,
0.26063148714913287,
0.9905681235090423,
0.5046687378016,
0.034801332146161545,
0.8338263293112319,
0.5360967198090165,

0.6537230965024035,
0.17157001746672373,
0.4831429750094415,
0.9020531828964224,
0.6753247292337247,
0.5174379298434469,
0.9890178705073216,
0.267159704422534,
0.8341785578000804,
0.9379817087628837,
0.000581491561003733,
0.6145275069936325,
0.5766814780136865,
0.7982150659319514,
0.15489457430404485,
0.9170112120131346,
0.29881084391877955,
0.3418087919347266,
0.3595543901684044,
0.3535797670065908,
0.7824110389110103,
0.02056952570440085,
0.6304400807676319,
0.6686674255798845,
0.025494421196195827,
0.38518565667992954,
0.5431425666869112,
0.06684637498458734,
0.07581036200512403,
0.45184587762294903,
0.6641873646129721,
0.5304360199308287,
0.008878099704334064,
0.8422277928935051,
0.7785610400503568,
0.12704454298206214,
0.7418660999262898,
0.18819711772231917,
0.9719975119827262,
0.48343920531553775,
0.22036961776249764,
0.3509608943321645,
0.7374667862790814,
0.6359377097680654,
0.8102169364162963,
0.6379259801804009,
0.4472521704026622,

0.03232635622086244,
0.39213465138933945,
0.13300299810515392,
0.8809055112938508,
0.8435065949063154,
0.029708598389061147,
0.4593908719342754,
0.5213414081992561,
0.22987078669817718,
0.3174393408712374,
0.5353052634869624,
0.8772758329676057,
0.8246761276881277,
0.3312004993602711,
0.9490302011012344,
0.0911446314916009,
0.16290909173375312,
0.6337339152310999,
0.8158814728517106,
0.8619712889599244,
0.05536807251205056,
0.0911285677352982,
0.5343152728360909,
0.17339162942606623,
0.7102384972496106,
0.7266075691526561,
0.026971506576608517,
0.7990120302994344,
0.8766591214587255,
0.9498405965312198,
0.2531825294605986,
0.835895346243271,
0.9743317526845885,
0.7414766152196057,
0.4839520252578915,
0.6563739798638792,
0.14525504774402864,
0.4148866141080533,
0.9526564386900949,
0.9644540541591017,
0.2209669810643914,
0.17133440351444562,
0.9323630053732717,
0.5065184755598526,
0.16898272551525428,
0.7607386384772649,
0.8892605419142345,

```
0.6597671106117673,
0.8838538359157184,
0.5045188625635538,
0.7250466835534146,
0.7197671421276478,
0.46276294540853147,
...]
```

[ ]:

```

```