

# random\_resistance\_strategy

October 30, 2020

## 1 Winning probability of a random strategy as proposed on page 24

```
[1]: from scipy.special import binom
```

```
[2]: N = 5 # number of players
R = 3 # number of resistance fighters
S = 2 # number of spies
M = [2, 3, 2, 3, 3] # mission sizes
m = len(M) # number of missions
```

$$P(\text{mission won by resistance}) = \sum_{s=0}^{\min\{S, \text{size}\}} \frac{\binom{S}{s} \cdot \binom{R}{\text{size}-s}}{\binom{N}{\text{size}}} \frac{1}{2^s}$$

```
[3]: def mission_success_prob (mission_size, verbose = False):
    p = 0
    for s in range (1 + min(S, mission_size)): # s is number of spies in mission
        # number of species in mission (s) has hypergeometric distribution
        ↪ (drawing without replacement)
        p_s = binom(S, s) * binom(R, mission_size - s) / binom(N, mission_size)
        p_success_given_s = .5 ** s # mission succeeds iff all s spies vote for
        ↪ success
        if verbose:
            print ("s=", s, "\tp_s=", p_s, "\tp_success_given_s=",
        ↪ p_success_given_s)
        p += p_s * p_success_given_s
    return p
```

```
[4]: mission_success_prob(2, True)
```

s= 0	p_s= 0.3	p_success_given_s= 1.0
s= 1	p_s= 0.6	p_success_given_s= 0.5
s= 2	p_s= 0.1	p_success_given_s= 0.25

```
[4]: 0.625
```

```
[5]: mission_success_prob(3, True)
```

```
s= 0    p_s= 0.1        p_success_given_s= 1.0
s= 1    p_s= 0.6        p_success_given_s= 0.5
s= 2    p_s= 0.3        p_success_given_s= 0.25
```

```
[5]: 0.47500000000000003
```

## 1.1 P(game is won)

$m$  independent Bernoulli trials with different success probabilities, game is won if at least 3 missions are won

```
[6]: winprob = 0
p = [None] * 5
a = [None] * 5
for outcome_idx in range(2**m): # loop over all outcome combinations of all
    ↳missions
    for j in range(m):
        a[j] = int(bool(outcome_idx & 1 << j)) # j-th bit
        p[j] = mission_success_prob(M[j])
    if sum(a) >= 3: # more than half of the 5 missions are successes
        q = 1.0
        for j in range(5):
            q *= p[j] ** a[j] * (1. - p[j]) ** (1. - a[j])
        print ("{:2d}-th outcome {} has probability {:.4f}".format(outcome_idx,
    ↳a, q))
        winprob += q

print ("\nProbability that resistance wins game is {:.5f}".format(winprob))
```

```
7-th outcome [1, 1, 1, 0, 0] has probability 0.0511
11-th outcome [1, 1, 0, 1, 0] has probability 0.0278
13-th outcome [1, 0, 1, 1, 0] has probability 0.0511
14-th outcome [0, 1, 1, 1, 0] has probability 0.0278
15-th outcome [1, 1, 1, 1, 0] has probability 0.0463
19-th outcome [1, 1, 0, 0, 1] has probability 0.0278
21-th outcome [1, 0, 1, 0, 1] has probability 0.0511
22-th outcome [0, 1, 1, 0, 1] has probability 0.0278
23-th outcome [1, 1, 1, 0, 1] has probability 0.0463
25-th outcome [1, 0, 0, 1, 1] has probability 0.0278
26-th outcome [0, 1, 0, 1, 1] has probability 0.0151
27-th outcome [1, 1, 0, 1, 1] has probability 0.0251
28-th outcome [0, 0, 1, 1, 1] has probability 0.0278
29-th outcome [1, 0, 1, 1, 1] has probability 0.0463
30-th outcome [0, 1, 1, 1, 1] has probability 0.0251
31-th outcome [1, 1, 1, 1, 1] has probability 0.0419
```

Probability that resistance wins game is 0.56598

[ ]: