Blucher

February 6, 2021

1 Blűcher

1.1 Introduction

[49]:
```
from functools import reduce
from math import comb, floor

def choose(n, k):
    '''math.comb with the convention that k < 0 is 0'''
    if k < 0:
        return 0
    else:
        return comb(n, k)
```

1.2 Fire

Fire combat is given by a number of dice (for an artillery unit, the current ammunition level of the unit; for an infantry unit, the current elan of the unit), possibly with a penalty or a bonus. Normally, every 6 rolled gives one hit; with the bonus *one* five rolled gives an additional hit. The number of dice may be halved due to penalties to firing.

For infantry and cavalry, each hit causes one fatigue. For artillery, 1 hit causes a retreat; more than one hit (during a single fire phase) causes the artillery unit to **retire.**

See Probability of i event_A and j event_B on n dice for the computations below.

[50]:
```
def fireRoll(n, bonus=False):
    result = [0] * (n + 1)
    for i in range(0, n + 1):
        # (number of ways of choosing i dice * succ^i * fail^(dice - i))
        if not bonus:
            result[i] = (
                choose(n, i)
                * pow(1, i)
                * pow(5, n - i)
            )
        else:
            pSixes    =  (
                choose(n, i)
```

```
                * pow(1, i)
                * pow(4, n - i)
            )
            pBonus    =  (
                choose(n, i-1)
                * pow(1, i-1)
                * (pow(5, n - (i-1)) - pow(4, n - (i-1)))
            )
            result[i] = (pSixes + pBonus)
        # finally, divide by 6^n
        result[i] = result[i] / pow(6, n)
    return result

def fire(n, bonus=False):
    exp = reduce(
        lambda acc, e: (acc + (e[0] * e[1])),
        enumerate(fireRoll(n, bonus)),
        0
    )
    return f'{floor((exp * 1000) + 0.5) / 1000}'
```

1.2.1 Artillery

This table gives the number of fire hits, based on the number of dice rolled.

Ammo (dice)	No bonus	bonus
6	1.0	1.665
5	0.833	1.431
4	0.667	1.184
3	0.5	0.921
2	0.333	0.639
1	0.167	0.333

1.2.2 Infantry

This table gives the number of fire hits, based on the number of dice rolled.

Elan (dice)	No bonus	bonus
8	1.333	2.101
7	1.167	1.888
6	1.0	1.665
5	0.833	1.431
4	0.667	1.184
3	0.5	0.921
2	0.333	0.639
1	0.167	0.333

For infantry and cavalry, each hit causes one fatigue. For artillery, 1 hit causes a retreat; more than one hit (during a single fire phase) causes the artillery unit to **retire.**

2 Combat

Combat in Blucher, as opposed to shooting, involves the attacker and defender each rolling some number of dice and counting successes, with the one with more successes being the winner. The defender wins ties. The number of dice is based on an attribute of the unit, either its elan or its current ammunition, for artillery, with modifiers adding and removing dice.

[51]:
```
def combatRoll(n, minSuccess = 4, rerolling = False):
    nSuccesses = 6 - minSuccess + 1
    result = [0.0] * (n + 1)
    for i in range(0, n + 1):
        result[i] = (
            choose(n, i)
            * pow(nSuccesses, i)
            * pow(6 - nSuccesses, n - i)
        ) / pow(6, n)
    if rerolling:
        reroll = [0.0] * (n + 1)
        reroll[0] = result[0]
        for i in range(1, n + 1):
            newRoll = combatRoll(i, minSuccess)
            for j in range(0, len(newRoll)):
                reroll[j] += result[i] * newRoll[j]
        result = reroll
    return result

def combat(n, minSuccess = 4, rerolling = False):
    exp = reduce(
        lambda acc, e: (acc + (e[0] * e[1])),
        enumerate(combatRoll(n, minSuccess, rerolling)),
        0
    )
    return f'{floor((exp * 1000) + 0.5) / 1000}'
```

Under some circumstances, primarily having to do with cavalry attacking or defending against infantry, either the attacker or defender's successes have to be rerolled.

The following table shows the expected results of rolling dice with and without rerolling successes.

Dice	Without rerolling	Rerolling hits
9	4.5	2.25
8	4.0	2.0
7	3.5	1.75
6	3.0	1.5
5	2.5	1.25

Dice	Without rerolling	Rerolling hits
4	2.0	1.0
3	1.5	0.75
2	1.0	0.5
1	0.5	0.25

[52]:
```
def combatResult(atk, dfs, minSuccess=4, atkReroll=False, dfsReroll=False):
    atkRoll = combatRoll(atk, rerolling=atkReroll)
    dfsRoll = combatRoll(dfs, rerolling=dfsReroll)
    atkWins = 0.0
    dfsWins = 0.0
    for a in range(0, len(atkRoll)):
        for d in range(0, len(dfsRoll)):
            if a <= d:
                dfsWins += atkRoll[a] * dfsRoll[d]
            else:
                atkWins += atkRoll[a] * dfsRoll[d]
    return [atkWins, dfsWins]

def r(atk, dfs, atkReroll=False, dfsReroll=False):
    def fmt(n): return floor((n * 100) + 0.5)
    res = combatResult(atk, dfs, atkReroll=atkReroll, dfsReroll=dfsReroll)
    res = list(map(fmt, res))
    return f'{res[0]}% / {res[1]}%'

def s(atk, dfs): return r(atk, dfs, atkReroll=True)
def t(atk, dfs): return r(atk, dfs, dfsReroll=True)
```

The next table shows the probability of attacker/defender winning based on the number of dice rolled. Attacker dice are along the left, defender dice are along the top. This table assumes neither rerolls successes.

ATK\DFS	2	3	4	5
2	31% / 69%	19% / 81%	11% / 89%	6% / 94%
3	50% / 50%	34% / 66%	23% / 77%	14% / 86%
4	66% / 34%	50% / 50%	36% / 64%	25% / 75%
5	77% / 23%	64% / 36%	50% / 50%	38% / 62%
6	86% / 14%	75% / 25%	62% / 38%	50% / 50%
7	91% / 9%	83% / 17%	73% / 27%	61% / 39%
8	95% / 5%	89% / 11%	81% / 19%	71% / 29%
9	97% / 3%	93% / 7%	87% / 13%	79% / 21%
10	98% / 2%	95% / 5%	91% / 9%	85% / 15%

ATK\DFS	6	7	8	9	10
2	4% / 96%	2% / 98%	1% / 99%	1% / 99%	0% / 100%
3	9% / 91%	5% / 95%	3% / 97%	2% / 98%	1% / 99%
4	17% / 83%	11% / 89%	7% / 93%	5% / 95%	3% / 97%
5	27% / 73%	19% / 81%	13% / 87%	9% / 91%	6% / 94%
6	39% / 61%	29% / 71%	21% / 79%	15% / 85%	11% / 89%
7	50% / 50%	40% / 60%	30% / 70%	23% / 77%	17% / 83%
8	60% / 40%	50% / 50%	40% / 60%	31% / 69%	24% / 76%
9	70% / 30%	60% / 40%	50% / 50%	41% / 59%	32% / 68%
10	77% / 23%	69% / 31%	59% / 41%	50% / 50%	41% / 59%

In general for cavalry and infantry, when the attacker wins, the attacker takes a fatigue and may advance while the defender takes the difference as fatigues and must retreat if it is not broken. When the defender wins, the attacker takes two fatigues and must retreat while the defender takes one fatigue.

When cavalry attacks infantry, or vice-versa, a number of different rules come into use. If the infantry is *prepared* (think of this as in a square formation), the attacking cavalry must reroll its successes. If the infantry is attacking, it cannot be prepared and must reroll successes.

This table assumes the attacker rerolls successes.

ATK\DFS	2	3	4	5
2	14% / 86%	8% / 92%	4% / 96%	2% / 98%
3	23% / 77%	14% / 86%	8% / 92%	5% / 95%
4	31% / 69%	20% / 80%	13% / 87%	8% / 92%
5	40% / 60%	27% / 73%	18% / 82%	12% / 88%
6	48% / 52%	35% / 65%	24% / 76%	16% / 84%
7	56% / 44%	42% / 58%	30% / 70%	21% / 79%
8	62% / 38%	48% / 52%	37% / 63%	27% / 73%
9	68% / 32%	55% / 45%	43% / 57%	32% / 68%
10	73% / 27%	61% / 39%	49% / 51%	38% / 62%

ATK\DFS	6	7	8	9	10
2	1% / 99%	1% / 99%	0% / 100%	0% / 100%	0% / 100%
3	3% / 97%	2% / 98%	1% / 99%	0% / 100%	0% / 100%
4	5% / 95%	3% / 97%	2% / 98%	1% / 99%	1% / 99%
5	8% / 92%	5% / 95%	3% / 97%	2% / 98%	1% / 99%
6	11% / 89%	7% / 93%	5% / 95%	3% / 97%	2% / 98%
7	15% / 85%	10% / 90%	7% / 93%	4% / 96%	3% / 97%
8	19% / 81%	13% / 87%	9% / 91%	6% / 94%	4% / 96%
9	24% / 76%	17% / 83%	12% / 88%	8% / 92%	6% / 94%
10	29% / 71%	21% / 79%	15% / 85%	11% / 89%	8% / 92%

On the other hand, cavalry attacking unprepared infantry forces the infantry to reroll successes.

This table assumes the defender rerolls successes.

ATK\DFS	2	3	4	5
2	52% / 48%	42% / 58%	34% / 66%	28% / 72%
3	69% / 31%	60% / 40%	51% / 49%	44% / 56%
4	80% / 20%	73% / 27%	66% / 34%	58% / 42%
5	88% / 12%	82% / 18%	76% / 24%	70% / 30%
6	93% / 7%	89% / 11%	84% / 16%	79% / 21%
7	96% / 4%	93% / 7%	90% / 10%	86% / 14%
8	98% / 2%	96% / 4%	93% / 7%	90% / 10%
9	99% / 1%	97% / 3%	96% / 4%	94% / 6%
10	99% / 1%	98% / 2%	97% / 3%	96% / 4%

ATK\DFS	6	7	8	9	10
2	22% / 78%	18% / 82%	14% / 86%	11% / 89%	9% / 91%
3	37% / 63%	31% / 69%	26% / 74%	22% / 78%	18% / 82%
4	51% / 49%	45% / 55%	39% / 61%	33% / 67%	29% / 71%
5	64% / 36%	57% / 43%	51% / 49%	45% / 55%	40% / 60%
6	74% / 26%	68% / 32%	62% / 38%	57% / 43%	51% / 49%
7	81% / 19%	77% / 23%	72% / 28%	66% / 34%	61% / 39%
8	87% / 13%	83% / 17%	79% / 21%	75% / 25%	70% / 30%
9	91% / 9%	88% / 12%	85% / 15%	81% / 19%	77% / 23%
10	94% / 6%	92% / 8%	89% / 11%	86% / 14%	83% / 17%

2.0.1 Artillery defending

When an artillery unit is attacked by a cavalry or infantry unit, if the cavalry or infantry gets twice as many, or more, successes than the artillery, the artillery is broken.

[53]:
```
def artilleryResult(atk, dfs, minSuccess=4):
    atkRoll = combatRoll(atk)
    dfsRoll = combatRoll(dfs)
    dfsWins = 0.0
    atkWins = 0.0
    atkDbl  = 0.0
    for a in range(0, len(atkRoll)):
        for d in range(0, len(dfsRoll)):
            if a <= d:
                dfsWins += atkRoll[a] * dfsRoll[d]
            elif a < 2*d:
                atkWins += atkRoll[a] * dfsRoll[d]
            else:
                atkDbl += atkRoll[a] * dfsRoll[d]
    return [atkWins, atkDbl, dfsWins]
```

```
def a(atk, dfs):
    def fmt(n): return floor((n * 100) + 0.5)
    result = list(map(fmt, artilleryResult(atk, dfs)))
    return f'{result[1]}% / {result[0]}%'
```

The next chart presents the chances of the artillery breaking and the chances of the artillery merely being forced to retreat based on the number of dice rolled. (The other option is the defending artillery winning.)

ATK\DFS	2	3	4	5
2	31% / 0%	19% / 0%	11% / 0%	6% / 0%
3	47% / 3%	30% / 5%	18% / 5%	11% / 4%
4	59% / 6%	40% / 10%	25% / 11%	16% / 10%
5	70% / 8%	50% / 14%	33% / 17%	22% / 16%
6	78% / 8%	59% / 16%	42% / 21%	28% / 22%
7	84% / 7%	67% / 16%	50% / 23%	35% / 26%
8	89% / 5%	74% / 14%	58% / 23%	43% / 28%
9	93% / 4%	80% / 12%	65% / 21%	50% / 29%
10	95% / 3%	85% / 10%	72% / 19%	57% / 28%

ATK\DFS	6	7	8	9	10
2	4% / 0%	2% / 0%	1% / 0%	1% / 0%	0% / 0%
3	6% / 3%	3% / 2%	2% / 1%	1% / 1%	1% / 1%
4	9% / 8%	6% / 6%	3% / 4%	2% / 3%	1% / 2%
5	14% / 14%	8% / 11%	5% / 8%	3% / 6%	2% / 4%
6	18% / 20%	12% / 17%	7% / 14%	4% / 11%	3% / 8%
7	24% / 26%	16% / 24%	10% / 20%	6% / 16%	4% / 13%
8	30% / 30%	21% / 29%	14% / 27%	9% / 23%	6% / 18%
9	37% / 33%	26% / 34%	18% / 32%	12% / 29%	8% / 25%
10	43% / 34%	32% / 37%	22% / 37%	15% / 35%	10% / 31%