In [285…

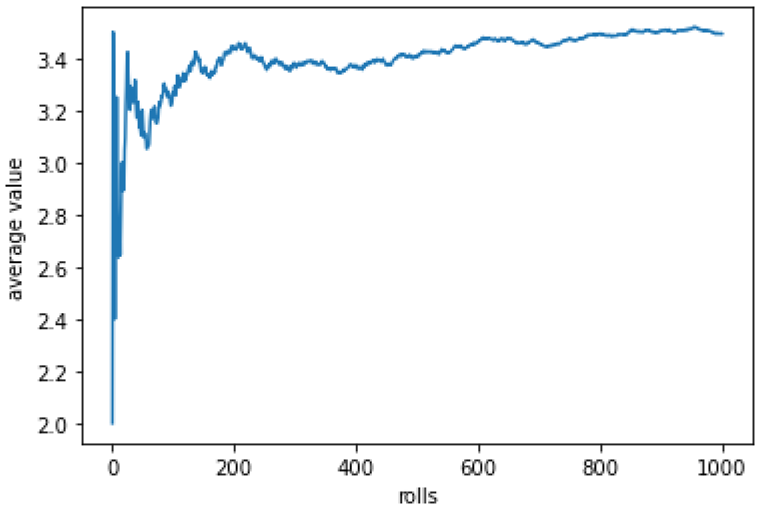
```python
from dataclasses import dataclass
from random import randint
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [52]:

```python
@dataclass
class Die:
    """
    Simple interface for rolling an n-sided die
    """
    n: int = 6

    def __call__(self) -> int:
        return randint(1, self.n)
```

In [56]:

```python
"""
Verify that the die works how we expect
"""
die = Die()

x = np.arange(1, 1001)
rolls = [die() for _ in x]

plt.plot(x, np.cumsum(rolls)/(x))
plt.xlabel("rolls")
plt.ylabel("average value")
plt.show()
```



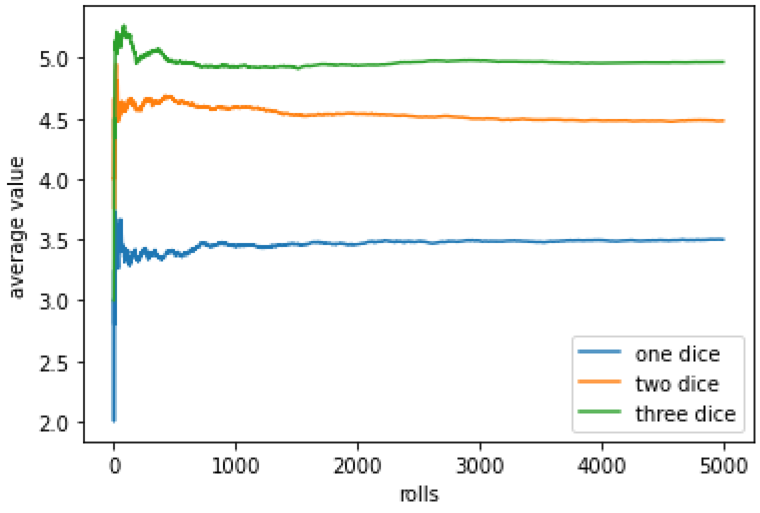
We see our EV is, as expected, about 3.5.

In [53]:

```python
@dataclass
class Dice:
    """
    Simple interface for a bunch of dice
    """
    dice: list[Die]

    def __call__(self) -> list[int]:
        """
        Returns three rolls in descending order
        """
        return sorted([die() for die in self.dice], reverse=True)
```

In [57]:

```
"""
Verify that rolling a bunch of dice and taking the highest
number works as we expect.
"""
two_dice = Dice([die, die])
three_dice = Dice([die, die, die])

x = np.arange(1, 5001)

one = [die() for _ in x]
two = [two_dice()[0] for _ in x]
three = [three_dice()[0] for _ in x]

plt.plot(x, np.cumsum(one)/(x), label="one dice")
plt.plot(x, np.cumsum(two)/(x), label="two dice")
plt.plot(x, np.cumsum(three)/(x), label="three dice")
plt.legend()
plt.xlabel("rolls")
plt.ylabel("average value")
plt.show()
```



We see our EV's follow the expected pattern, 5 for 3 dice, 4.5 for 2 dice, 3.5 for 1 die.

In [201…

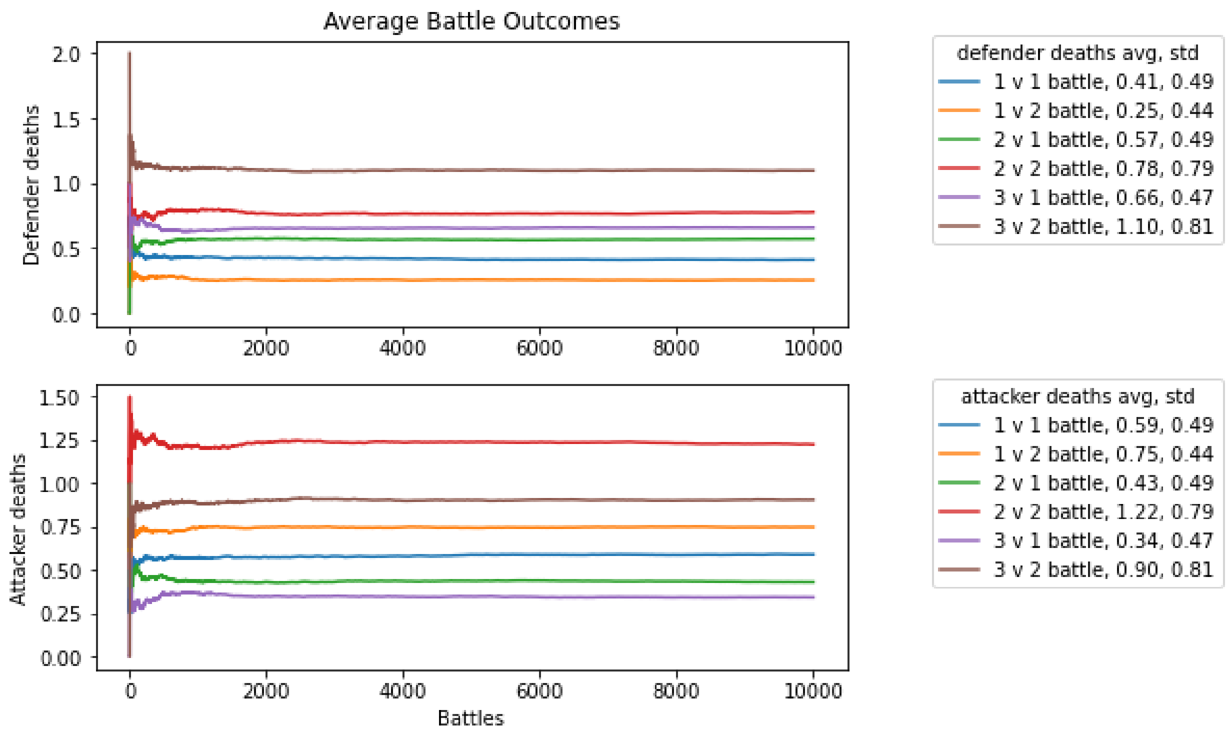
```
@dataclass
class Battle:
    """
    Battle two sets of dice and give the outcome for each matched pair of dice
    (attacker vs defender), True if attacker gets the kill and False if attacke
    is killed.
    """
    attackers: Dice
    defenders: Dice

    def __post_init__(self):
        self.n_kills = min(len(self.attackers.dice), len(self.defenders.dice))

    def __call__(self) -> list[bool]:
        return [
            attacker > defender
            for attacker, defender in zip(
                self.attackers(), self.defenders()
            )
        ][:self.n_kills]
```

In [191…

```
x = np.arange(1, 10001)
fig, [def_ax, atk_ax] = plt.subplots(2, figsize=(7, 6))

for n_atk in range(1, 4):
    for n_def in range(1, 3):
        battle = Battle(Dice([die] * n_atk), Dice([die] * n_def))
        n_kills = min(n_atk, n_def)
        def_deaths = [sum(battle()) for _ in x]
        atk_deaths = [n_kills - def_death for def_death in def_deaths]
        def_ax.plot(
            x,
            np.cumsum(def_deaths)/(x),
            label=f"{n_atk} v {n_def} battle, {np.mean(def_deaths):.2f}, {np.s
        )
        atk_ax.plot(
            x,
            np.cumsum(atk_deaths)/(x),
            label=f"{n_atk} v {n_def} battle, {np.mean(atk_deaths):.2f}, {np.s
        )

def_ax.set_title("Average Battle Outcomes")
atk_ax.legend(bbox_to_anchor=(1.1, 1.05), title="attacker deaths avg, std")
def_ax.legend(bbox_to_anchor=(1.1, 1.05), title="defender deaths avg, std")
atk_ax.set_ylabel("Attacker deaths")
def_ax.set_ylabel("Defender deaths")
atk_ax.set_xlabel("Battles")
```

Out[191]:

```
Text(0.5, 0, 'Battles')
```



Here we see the outcome for each kind of battle possible in the game. The numbers generally make sense and scale in reasonable ways.

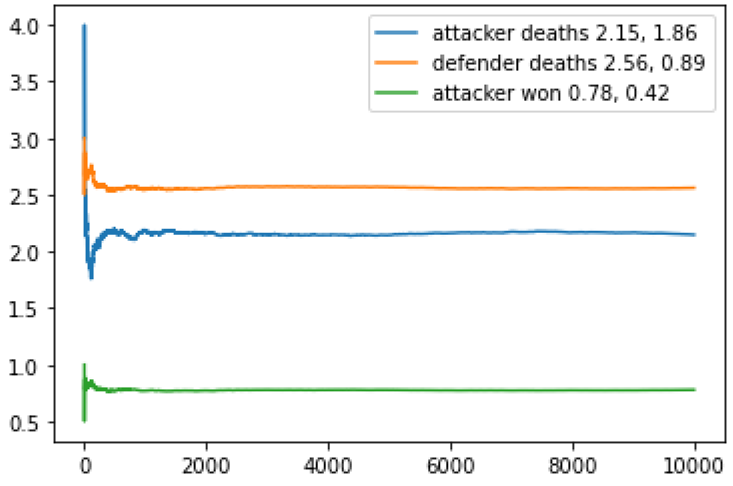
```
1 v 1: defender is favored by ~1/6, i.e. tie advantage
2 v 1: defender gets the kill 43% of the time
2 v 2: defender advantage is higher than it is in 1 v 1
3 v 1: clutch factor, defender still wins 1/3 the time
3 v 2: kind of the big point to make here, attacker is slightly
favored
```

In [367…

```
@dataclass
class StandardWarOfAttrition:
    """
    Attacker begins a war of attrition with some number of units. Staging for
    the battle is reset after each battle, following the standard rules.
    """
    attacker_units: int
    defender_units: int

    def __post_init__(self):
        self.battle_dict = {}
        for a in range(1, 4):
            for b in range(1, 3):
                self.battle_dict[f"{a}{b}"] = Battle(
                    Dice([die] * a),
                    Dice([die] * b)
                )

    def __call__(self):
        remaining_attackers = self.attacker_units
        remaining_defenders = self.defender_units
        while remaining_attackers and remaining_defenders:
            battle = self.battle_dict[
                f"{min(3, remaining_attackers)}{min(2, remaining_defenders)}"
            ]()
            remaining_defenders -= sum(battle)
            remaining_attackers -= len(battle) - sum(battle)
        return (
            self.attacker_units - remaining_attackers,
            self.defender_units - remaining_defenders,
            remaining_attackers > 0
        )
```

In [368…

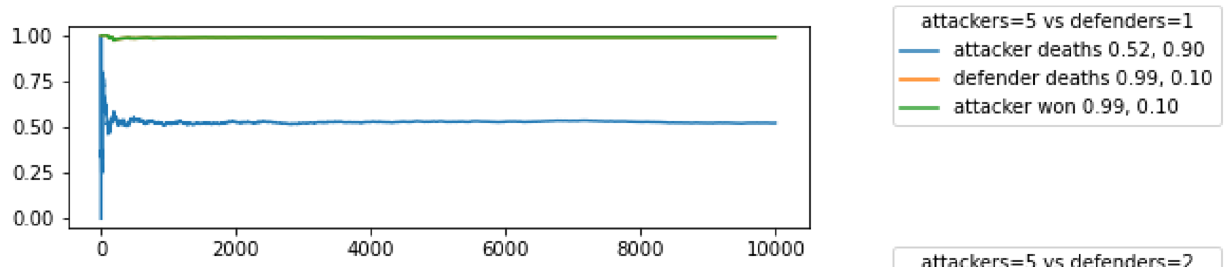
```
five_v_three = StandardWarOfAttrition(5, 3)

x = np.arange(1, 10000)
outcomes = [five_v_three() for _ in x]
atk_deaths = [out[0] for out in outcomes]
def_deaths = [out[1] for out in outcomes]
atk_won = [out[2] for out in outcomes]
plt.plot(
    x,
    np.cumsum(atk_deaths)/(x),
    label=f"attacker deaths {np.mean(atk_deaths):.2f}, {np.std(atk_deaths):.2f
)
plt.plot(
    x,
    np.cumsum(def_deaths)/(x),
    label=f"defender deaths {np.mean(def_deaths):.2f}, {np.std(def_deaths):.2f
)
plt.plot(
    x,
    np.cumsum(atk_won)/(x),
    label=f"attacker won {np.mean(atk_won):.2f}, {np.std(atk_won):.2f}"
)
plt.legend()
plt.show()
```

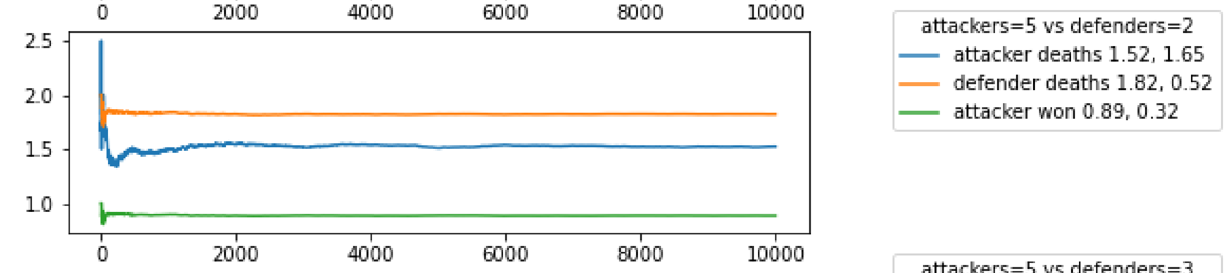


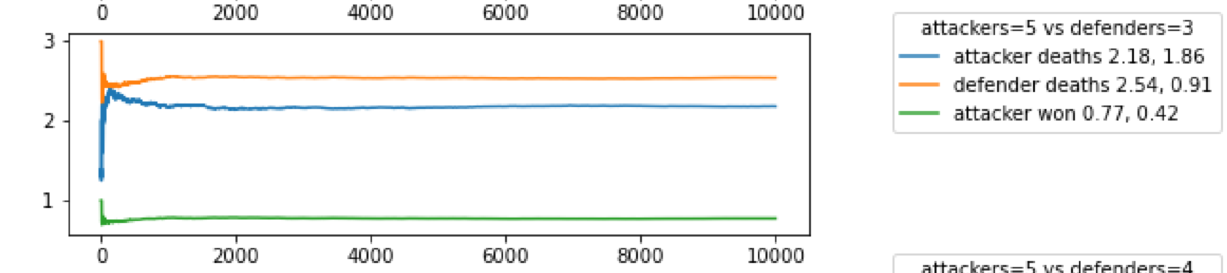
Here we get the first look at what happens on average in wars of attrition, using the standard rules. In a 5 vs 3, the attacker usually wins, whaddya know! EV is 0.77, 77% chance of a win. That said, the standard deviation is %42!! Huge!

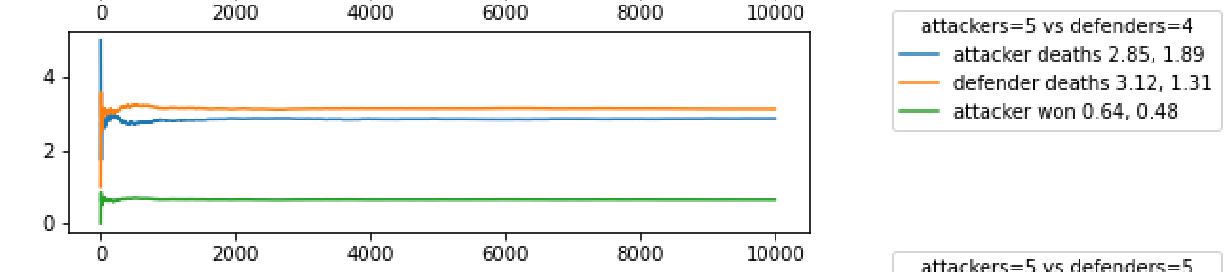
In [370…

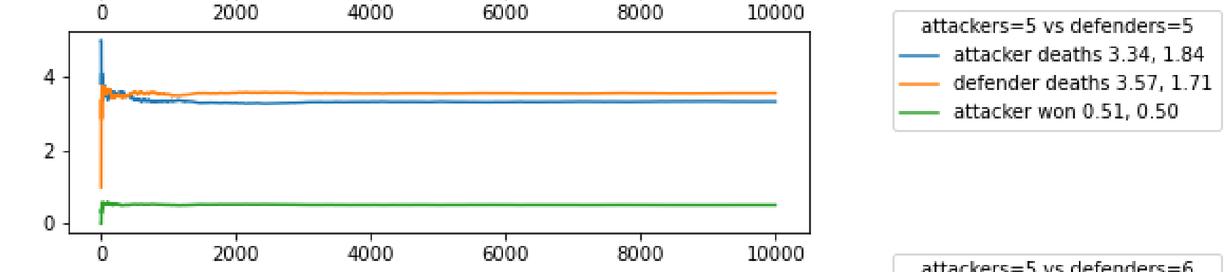
```
defenders = 7
fig, axs = plt.subplots(defenders, figsize=(7, 16))
for defenders, ax in zip(range(1, defenders + 1), axs):
    five_v_x = StandardWarOfAttrition(5, defenders)

    x = np.arange(1, 10000)
    outcomes = [five_v_x() for _ in x]
    atk_deaths = [out[0] for out in outcomes]
    def_deaths = [out[1] for out in outcomes]
    atk_won = [out[2] for out in outcomes]
    ax.plot(
        x,
        np.cumsum(atk_deaths)/(x),
        label=f"attacker deaths {np.mean(atk_deaths):.2f}, {np.std(atk_deaths)
    )
    ax.plot(
        x,
        np.cumsum(def_deaths)/(x),
        label=f"defender deaths {np.mean(def_deaths):.2f}, {np.std(def_deaths)
    )
    ax.plot(
        x,
        np.cumsum(atk_won)/(x),
        label=f"attacker won {np.mean(atk_won):.2f}, {np.std(atk_won):.2f}"
    )
    ax.legend(bbox_to_anchor=(1.1, 1.15), title=f"attackers=5 vs {defenders=}"
plt.show()
```

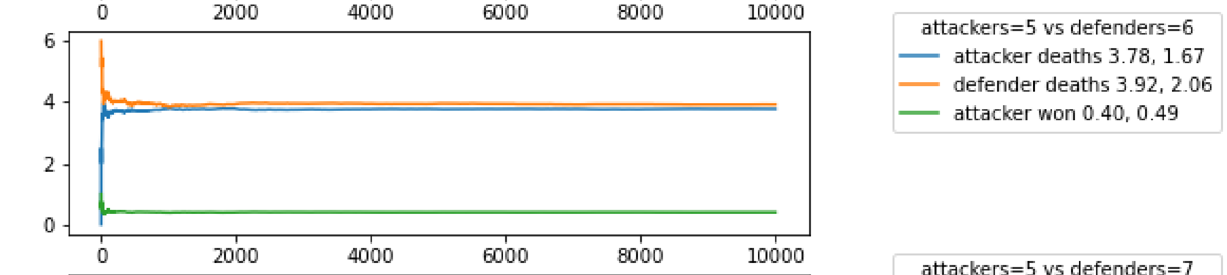


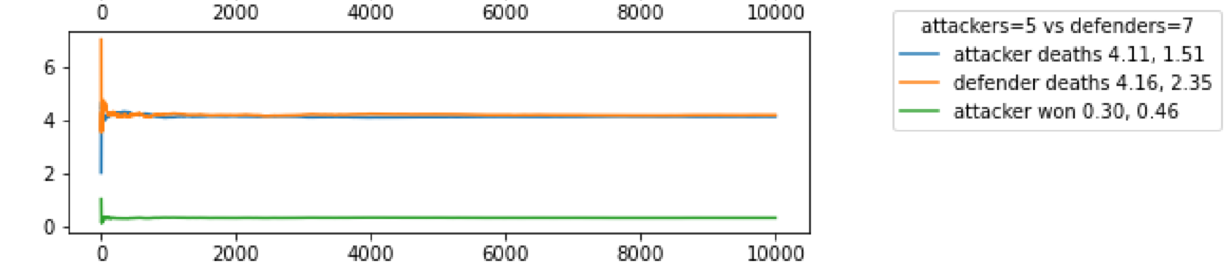












Here we're looking for the expected outcome with 5 attackers and n defenders, going from 1 to 7. To make sure things make sense and add up, let's look at the first plot briefly. The plot claims that on average, in a 5 v 1, ~0.54 attackers die. However, in a 3 v 1 (the first battle in the war of attrition), we can reference above and find that 0.34 attackers die? What gives?

Well,

$$0.58 \approx 0.34 + 0.34 \cdot 0.43 + 0.34 \cdot 0.45 \cdot 0.41 + 0.34 \cdot 0.45 \cdot 0.41 \cdot 0.41$$

if we add the chances that the *next* battle results in one attacker death, on and on, where one attacker dies 43% of the time in a 2 v 1 and 41% of the time in a 1 v 1. Considering the standard deviation is ~0.54 attackers die, +/- 0.9 attackers, the outcome is relatively close.

Further notes, obviously the chance that the attacker wins goes down each time you add a defender, usually by around 11% per defender. We can see in the 5 v 5 that the attacker is favored slightly, but variance is very high.

In a 5 v 7 we see that the same number of attackers and defenders die on average, even though the defenders still win 3/10 of the time.

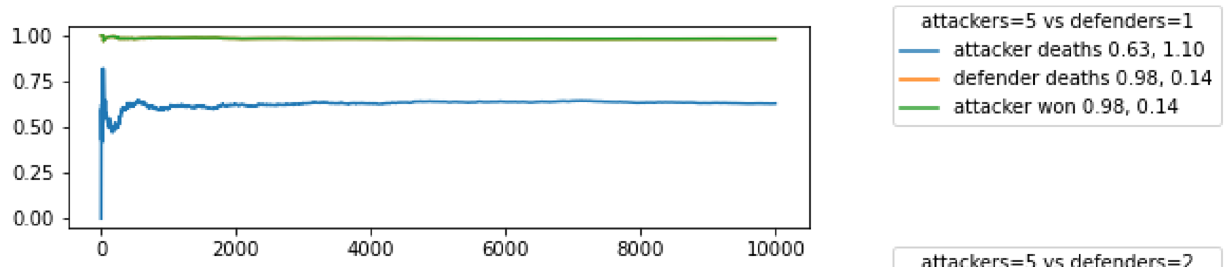
In [339…

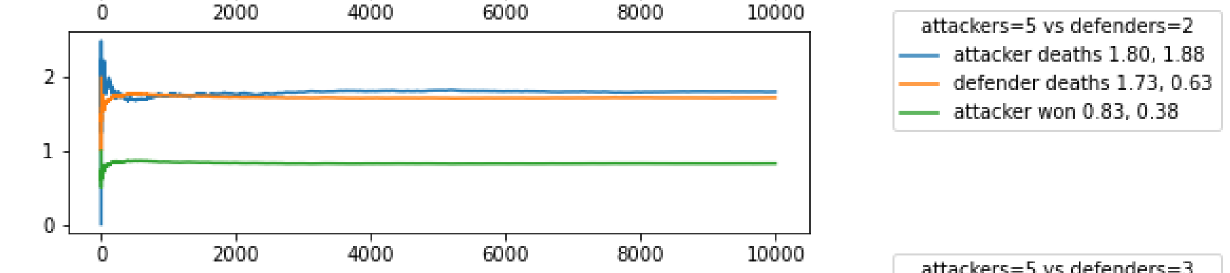
```
@dataclass
class TableWarOfAttrition:
    """
    Attacker begins a war of attrition with some number of units. Staging for
    the battle is only reset after either staged party is wiped, following odd
    table rules.
    """
    attacker_units: int
    defender_units: int

    def __post_init__(self):
        self.battle_dict = {}
        for a in range(1, 4):
            for b in range(1, 3):
                self.battle_dict[f"{a}{b}"] = Battle(
                    Dice([die] * a),
                    Dice([die] * b)
                )

    def __call__(self):
        remaining_attackers = self.attacker_units
        remaining_defenders = self.defender_units
        while remaining_attackers and remaining_defenders:
            staged_attackers = min(3, remaining_attackers)
            staged_defenders = min(2, remaining_defenders)
            while staged_attackers and staged_defenders:
                battle = self.battle_dict[
                    f"{staged_attackers}{staged_defenders}"
                ]()
                remaining_defenders -= sum(battle)
                remaining_attackers -= len(battle) - sum(battle)
                staged_defenders -= sum(battle)
                staged_attackers -= len(battle) - sum(battle)
        return (
            self.attacker_units - remaining_attackers,
            self.defender_units - remaining_defenders,
            remaining_attackers > 0
        )
```

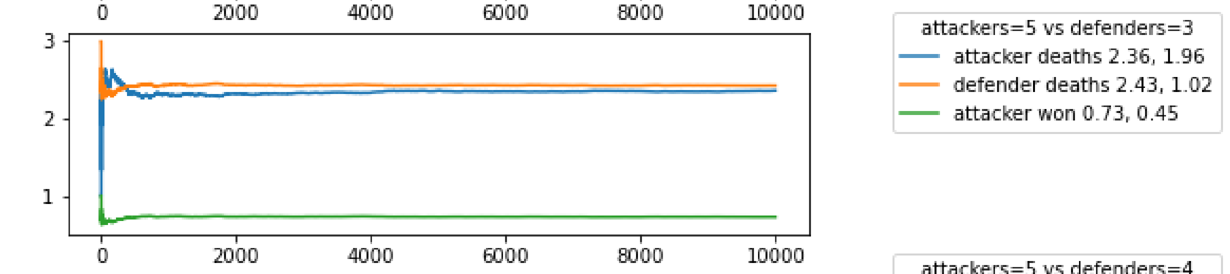
In [371…

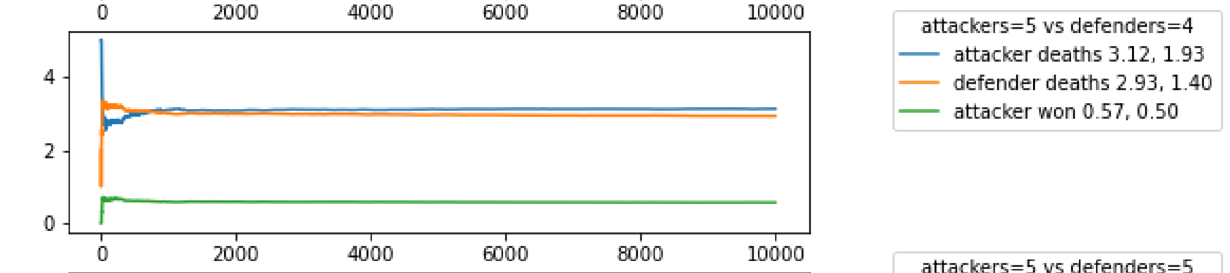
```
defenders = 7
fig, axs = plt.subplots(defenders, figsize=(7, 16))
```

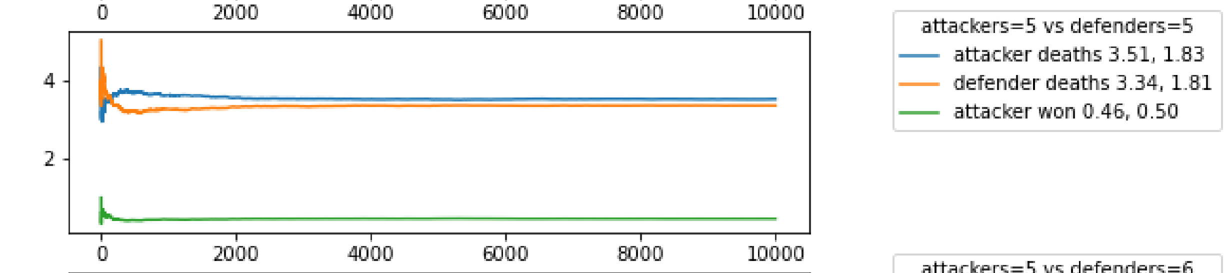
```
for defenders, ax in zip(range(1, defenders + 1), axs):
    five_v_x = TableWarOfAttrition(5, defenders)

    x = np.arange(1, 10000)
    outcomes = [five_v_x() for _ in x]
    atk_deaths = [out[0] for out in outcomes]
    def_deaths = [out[1] for out in outcomes]
    atk_won = [out[2] for out in outcomes]
    ax.plot(
        x,
        np.cumsum(atk_deaths)/(x),
        label=f"attacker deaths {np.mean(atk_deaths):.2f}, {np.std(atk_deaths)
    )
    ax.plot(
        x,
        np.cumsum(def_deaths)/(x),
        label=f"defender deaths {np.mean(def_deaths):.2f}, {np.std(def_deaths)
    )
    ax.plot(
        x,
        np.cumsum(atk_won)/(x),
        label=f"attacker won {np.mean(atk_won):.2f}, {np.std(atk_won):.2f}"
    )
    ax.legend(bbox_to_anchor=(1.1, 1.15), title=f"attackers=5 vs {defenders=}"
plt.show()
```

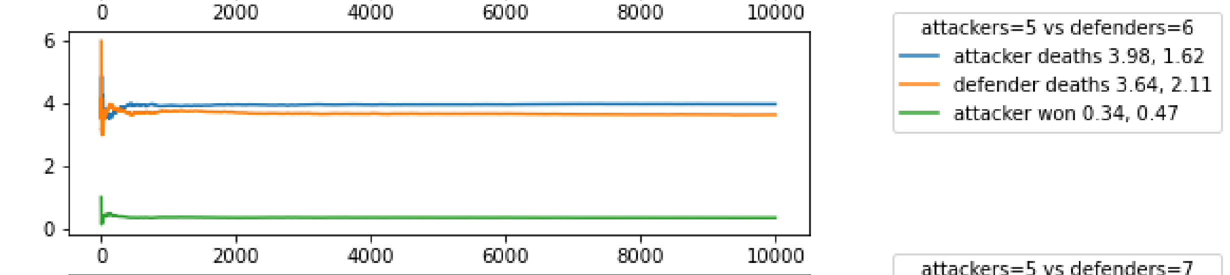


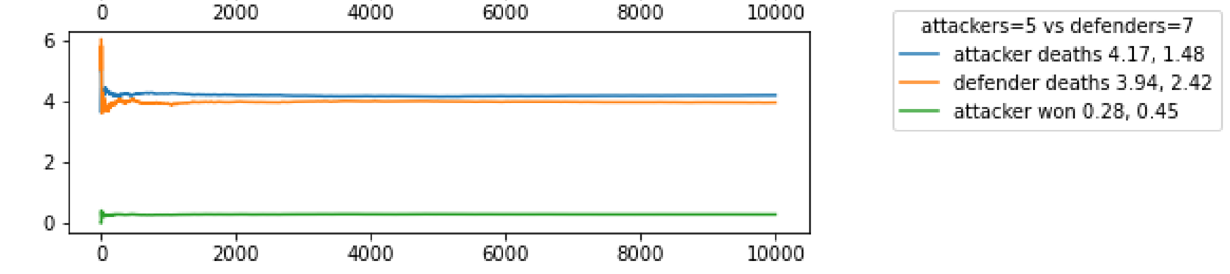












We can see that things changed, but it's hard to compare. Lets look more closely at how these six values - attacker deaths and variance, defender deaths and variance, attacker won and variance - change when you go from standard to table rules.

In [300…

```
@dataclass
class AttritionEvaluator:
    n_iterations: int

    def __call__(
        self,
        war_of_attrition: StandardWarOfAttrition | TableWarOfAttrition
    ):
        return [war_of_attrition() for _ in range(self.n_iterations)]
```

In [381…

```
evaluator = AttritionEvaluator(50000)

standard_results = pd.DataFrame(
    {
        attackers: {
            defenders: evaluator(StandardWarOfAttrition(attackers, defenders))
            for defenders in range(1, 13)}
        for attackers in range(1, 13)
    },
)
table_results = pd.DataFrame(
    {
        attackers: {
            defenders: evaluator(TableWarOfAttrition(attackers, defenders))
            for defenders in range(1, 13)}
        for attackers in range(1, 13)
    },
)
```

In [382…

```
table_results.axes[1].name = "attackers"
table_results.axes[0].name = "defenders"
standard_results.axes[1].name = "attackers"
standard_results.axes[0].name = "defenders"
```

In [383…

```
def attacker_deaths_mean(outcomes):
    return np.mean([out[0] for out in outcomes])

def attacker_deaths_std(outcomes):
    return np.std([out[0] for out in outcomes])

def defender_deaths_mean(outcomes):
    return np.mean([out[1] for out in outcomes])

def defender_deaths_std(outcomes):
    return np.std([out[1] for out in outcomes])

def attacker_wins_mean(outcomes):
    return np.mean([out[2] for out in outcomes]) * 100

def attacker_wins_std(outcomes):
    return np.std([out[2] for out in outcomes]) * 100
```

In [ ]:

```
def make_pretty(styler):
    styler.background_gradient(axis=None, cmap="YlGnBu")
    styler.format(precision=2)
    return styler

for func in [
    attacker_deaths_mean,
    attacker_deaths_std,
    defender_deaths_mean,
    defender_deaths_std,
    attacker_wins_mean,
    attacker_wins_std,
]:
    print(f"{func.__name__}, standard")
    func_standard = standard_results.applymap(func)
    func_table = table_results.applymap(func)
    display(func_standard.style.pipe(make_pretty))
    print(f"{func.__name__}, table")
    display(func_table.style.pipe(make_pretty))
    print(f"{func.__name__}, difference")
    display((func_table-func_standard).style.pipe(make_pretty))
```

```
attacker_deaths_mean, standard
```

attackers	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
defenders												
1	0.59	0.67	0.57	0.54	0.52	0.53	0.52	0.53	0.52	0.51	0.52	0.52
2	0.89	1.41	1.41	1.54	1.50	1.55	1.53	1.55	1.54	1.56	1.54	1.56
3	0.97	1.67	1.88	2.08	2.18	2.24	2.26	2.28	2.31	2.29	2.30	2.29
4	0.99	1.86	2.29	2.67	2.84	3.01	3.06	3.15	3.17	3.21	3.21	3.19
5	1.00	1.92	2.54	3.02	3.37	3.59	3.75	3.86	3.91	3.95	4.02	3.99
6	1.00	1.97	2.72	3.35	3.79	4.16	4.38	4.56	4.64	4.74	4.80	4.85
7	1.00	1.98	2.82	3.53	4.13	4.57	4.92	5.16	5.36	5.48	5.59	5.65
8	1.00	1.99	2.88	3.69	4.35	4.92	5.36	5.70	5.95	6.14	6.29	6.40
9	1.00	2.00	2.93	3.78	4.55	5.19	5.72	6.13	6.51	6.75	6.95	7.12
10	1.00	2.00	2.96	3.86	4.69	5.41	6.02	6.55	6.98	7.31	7.56	7.80
11	1.00	2.00	2.97	3.91	4.78	5.56	6.25	6.86	7.37	7.76	8.10	8.34
12	1.00	2.00	2.98	3.94	4.85	5.69	6.44	7.12	7.71	8.19	8.60	8.91

```
attacker_deaths_mean, table
```

attackers	1	2	3	4	5	6	7	8	9	10	11	12
defenders												
1	0.58	0.66	0.57	0.62	0.63	0.61	0.62	0.61	0.61	0.63	0.62	0.62
2	0.90	1.41	1.42	1.67	1.79	1.78	1.85	1.88	1.86	1.87	1.89	1.90
3	0.96	1.64	1.79	2.11	2.33	2.37	2.49	2.55	2.58	2.59	2.59	2.62
4	0.99	1.85	2.28	2.76	3.14	3.28	3.48	3.58	3.66	3.73	3.75	3.77
5	1.00	1.91	2.49	3.04	3.50	3.76	4.01	4.22	4.31	4.43	4.48	4.52
6	1.00	1.97	2.71	3.39	3.98	4.39	4.75	5.03	5.20	5.34	5.47	5.53
7	1.00	1.98	2.78	3.54	4.21	4.71	5.12	5.50	5.76	5.98	6.15	6.27
8	1.00	1.99	2.88	3.71	4.48	5.09	5.63	6.10	6.46	6.71	6.97	7.17
9	1.00	2.00	2.92	3.78	4.60	5.29	5.91	6.45	6.86	7.27	7.53	7.80
10	1.00	2.00	2.95	3.87	4.74	5.52	6.21	6.83	7.38	7.78	8.18	8.46
11	1.00	2.00	2.97	3.90	4.80	5.63	6.38	7.06	7.68	8.20	8.62	9.03
12	1.00	2.00	2.98	3.94	4.88	5.75	6.56	7.32	7.99	8.61	9.10	9.57

```
attacker_deaths_mean, difference
```

attackers	1	2	3	4	5	6	7	8	9	10	11	12
defenders												
1	-0.00	-0.01	-0.00	0.09	0.11	0.08	0.10	0.08	0.09	0.12	0.10	0.10
2	0.00	0.01	0.01	0.13	0.29	0.23	0.31	0.33	0.32	0.31	0.35	0.34
3	-0.02	-0.02	-0.09	0.04	0.15	0.14	0.23	0.27	0.27	0.29	0.28	0.33
4	-0.00	-0.01	-0.01	0.09	0.30	0.27	0.42	0.44	0.50	0.52	0.54	0.58
5	-0.00	-0.01	-0.05	0.02	0.13	0.17	0.26	0.36	0.40	0.48	0.45	0.54
6	-0.00	0.00	-0.01	0.04	0.19	0.24	0.36	0.47	0.56	0.60	0.67	0.68
7	-0.00	-0.00	-0.03	0.00	0.07	0.15	0.21	0.34	0.39	0.50	0.56	0.62
8	-0.00	0.00	0.00	0.02	0.12	0.18	0.28	0.40	0.50	0.57	0.67	0.77
9	-0.00	-0.00	-0.01	-0.00	0.05	0.10	0.19	0.32	0.35	0.51	0.58	0.69
10	0.00	0.00	-0.00	0.01	0.05	0.12	0.19	0.28	0.40	0.47	0.62	0.66
11	0.00	-0.00	-0.00	-0.01	0.02	0.06	0.13	0.20	0.31	0.43	0.52	0.69
12	0.00	-0.00	0.00	0.00	0.03	0.06	0.12	0.20	0.28	0.42	0.50	0.66

```
attacker_deaths_std, standard
```

attackers	1	2	3	4	5	6	7	8	9	10	11	12
defenders												
1	0.49	0.85	0.94	0.93	0.90	0.90	0.89	0.90	0.90	0.88	0.88	0.88
2	0.31	0.84	1.29	1.55	1.63	1.73	1.75	1.80	1.79	1.82	1.80	1.82
3	0.16	0.69	1.29	1.63	1.87	2.03	2.13	2.20	2.25	2.25	2.28	2.27
4	0.08	0.47	1.13	1.56	1.88	2.15	2.31	2.46	2.54	2.62	2.65	2.67
5	0.04	0.37	0.97	1.44	1.83	2.15	2.41	2.60	2.76	2.83	2.95	2.97
6	0.02	0.24	0.78	1.23	1.67	2.05	2.37	2.64	2.83	2.99	3.11	3.21
7	0.01	0.18	0.65	1.07	1.49	1.91	2.27	2.59	2.86	3.05	3.24	3.35
8	0.00	0.12	0.52	0.89	1.31	1.72	2.12	2.47	2.79	3.06	3.25	3.45
9	0.00	0.09	0.41	0.75	1.13	1.53	1.94	2.33	2.67	2.97	3.24	3.47
10	0.00	0.05	0.32	0.61	0.94	1.34	1.73	2.13	2.50	2.84	3.17	3.43
11	0.00	0.04	0.25	0.50	0.80	1.15	1.55	1.92	2.32	2.70	3.03	3.35
12	0.00	0.02	0.20	0.41	0.67	0.98	1.35	1.73	2.11	2.50	2.86	3.21

`attacker_deaths_std, table`

attackers	1	2	3	4	5	6	7	8	9	10	11	12
defenders												
1	0.49	0.84	0.94	1.08	1.11	1.09	1.11	1.11	1.11	1.13	1.12	1.12
2	0.31	0.83	1.29	1.65	1.87	1.92	2.07	2.14	2.15	2.19	2.23	2.23
3	0.20	0.70	1.27	1.65	1.96	2.10	2.31	2.45	2.50	2.57	2.63	2.64
4	0.11	0.48	1.13	1.56	1.94	2.19	2.47	2.67	2.80	2.93	3.01	3.07
5	0.07	0.37	0.99	1.43	1.84	2.17	2.48	2.76	2.94	3.11	3.25	3.35
6	0.04	0.23	0.79	1.20	1.61	1.99	2.36	2.68	2.94	3.17	3.36	3.51
7	0.02	0.19	0.68	1.07	1.46	1.87	2.26	2.61	2.92	3.20	3.44	3.63
8	0.01	0.11	0.51	0.86	1.22	1.62	2.01	2.39	2.75	3.06	3.36	3.60
9	0.01	0.08	0.43	0.75	1.08	1.47	1.86	2.24	2.63	2.96	3.29	3.57
10	0.00	0.05	0.33	0.59	0.87	1.22	1.60	1.98	2.36	2.76	3.09	3.42
11	0.00	0.04	0.27	0.52	0.77	1.09	1.45	1.83	2.19	2.57	2.95	3.30
12	0.00	0.03	0.19	0.40	0.60	0.90	1.22	1.56	1.94	2.29	2.71	3.05

`attacker_deaths_std, difference`

attackers	1	2	3	4	5	6	7	8	9	10	11	12
defenders												
1	0.00	-0.00	0.00	0.15	0.21	0.19	0.22	0.21	0.21	0.26	0.24	0.24
2	-0.01	-0.01	0.00	0.10	0.24	0.18	0.33	0.35	0.37	0.38	0.43	0.41
3	0.04	0.01	-0.02	0.02	0.09	0.08	0.18	0.25	0.25	0.32	0.35	0.37
4	0.03	0.01	-0.00	0.00	0.05	0.04	0.15	0.21	0.27	0.32	0.36	0.41
5	0.02	0.01	0.02	-0.01	0.01	0.02	0.07	0.16	0.17	0.28	0.31	0.38
6	0.02	-0.00	0.01	-0.02	-0.05	-0.05	-0.00	0.04	0.11	0.18	0.26	0.29
7	0.01	0.01	0.03	-0.01	-0.03	-0.04	-0.01	0.02	0.06	0.14	0.20	0.29
8	0.00	-0.01	-0.01	-0.02	-0.09	-0.09	-0.11	-0.08	-0.04	0.01	0.11	0.16
9	0.01	-0.00	0.02	0.00	-0.05	-0.06	-0.09	-0.09	-0.04	-0.02	0.05	0.11
10	0.00	-0.00	0.01	-0.02	-0.06	-0.12	-0.13	-0.15	-0.13	-0.08	-0.07	-0.01
11	0.00	0.00	0.02	0.02	-0.03	-0.06	-0.11	-0.10	-0.13	-0.13	-0.08	-0.04
12	0.00	0.01	-0.01	-0.01	-0.07	-0.09	-0.13	-0.17	-0.17	-0.21	-0.15	-0.16

```
defender_deaths_mean, standard
```

attackers	1	2	3	4	5	6	7	8	9	10	11	12
defenders												
1	0.41	0.75	0.91	0.97	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2	0.37	0.92	1.44	1.65	1.83	1.90	1.95	1.97	1.99	1.99	2.00	2.00
3	0.35	1.06	1.82	2.24	2.54	2.72	2.83	2.90	2.94	2.97	2.98	2.99
4	0.35	1.10	2.07	2.62	3.12	3.41	3.63	3.75	3.85	3.90	3.94	3.96
5	0.34	1.14	2.23	2.94	3.55	4.00	4.32	4.54	4.70	4.81	4.87	4.92
6	0.34	1.14	2.32	3.12	3.91	4.45	4.91	5.23	5.49	5.65	5.76	5.84
7	0.34	1.15	2.39	3.28	4.13	4.84	5.39	5.82	6.15	6.41	6.58	6.71
8	0.34	1.16	2.45	3.39	4.37	5.13	5.79	6.34	6.77	7.10	7.35	7.54
9	0.35	1.15	2.45	3.47	4.47	5.35	6.12	6.77	7.29	7.71	8.05	8.29
10	0.35	1.16	2.47	3.51	4.56	5.52	6.36	7.11	7.70	8.24	8.66	8.96
11	0.34	1.15	2.49	3.54	4.64	5.64	6.57	7.38	8.07	8.71	9.22	9.62
12	0.34	1.16	2.48	3.56	4.71	5.71	6.71	7.58	8.39	9.09	9.67	10.18

```
defender_deaths_mean, table
```

attackers	1	2	3	4	5	6	7	8	9	10	11	12
defenders												
1	0.42	0.76	0.91	0.95	0.98	0.99	1.00	1.00	1.00	1.00	1.00	1.00
2	0.36	0.91	1.43	1.57	1.73	1.87	1.90	1.94	1.97	1.98	1.99	1.99
3	0.40	1.14	1.97	2.18	2.44	2.69	2.76	2.84	2.91	2.94	2.96	2.98
4	0.40	1.17	2.19	2.52	2.91	3.34	3.48	3.65	3.78	3.84	3.89	3.93
5	0.40	1.23	2.43	2.86	3.35	3.91	4.15	4.37	4.59	4.70	4.78	4.86
6	0.40	1.23	2.51	3.02	3.64	4.33	4.66	4.97	5.30	5.47	5.61	5.73
7	0.40	1.26	2.62	3.23	3.89	4.68	5.16	5.54	5.94	6.17	6.37	6.56
8	0.40	1.24	2.65	3.31	4.04	4.94	5.47	5.95	6.46	6.80	7.06	7.30
9	0.40	1.26	2.70	3.42	4.19	5.16	5.76	6.33	6.93	7.32	7.67	7.99
10	0.41	1.26	2.68	3.43	4.23	5.26	5.98	6.63	7.28	7.79	8.20	8.61
11	0.40	1.26	2.71	3.47	4.32	5.41	6.13	6.86	7.59	8.17	8.69	9.12
12	0.40	1.25	2.73	3.48	4.33	5.48	6.26	7.01	7.83	8.45	9.06	9.58

`defender_deaths_mean, difference`

attackers	1	2	3	4	5	6	7	8	9	10	11	12
defenders												
1	0.00	0.00	-0.00	-0.02	-0.01	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00
2	-0.01	-0.00	-0.00	-0.08	-0.10	-0.03	-0.05	-0.03	-0.02	-0.01	-0.01	-0.00
3	0.06	0.08	0.15	-0.06	-0.10	-0.02	-0.07	-0.06	-0.03	-0.03	-0.02	-0.01
4	0.05	0.07	0.12	-0.10	-0.21	-0.07	-0.15	-0.10	-0.07	-0.07	-0.05	-0.03
5	0.06	0.09	0.20	-0.08	-0.20	-0.09	-0.17	-0.18	-0.10	-0.11	-0.09	-0.06
6	0.06	0.09	0.19	-0.10	-0.27	-0.12	-0.24	-0.25	-0.19	-0.18	-0.15	-0.10
7	0.06	0.11	0.23	-0.06	-0.25	-0.16	-0.23	-0.28	-0.22	-0.24	-0.21	-0.15
8	0.06	0.08	0.19	-0.07	-0.33	-0.20	-0.32	-0.39	-0.31	-0.30	-0.29	-0.24
9	0.06	0.11	0.25	-0.05	-0.29	-0.19	-0.36	-0.44	-0.35	-0.39	-0.37	-0.30
10	0.06	0.10	0.21	-0.07	-0.33	-0.26	-0.39	-0.48	-0.42	-0.45	-0.45	-0.35
11	0.06	0.11	0.22	-0.07	-0.32	-0.24	-0.44	-0.52	-0.49	-0.54	-0.53	-0.50
12	0.06	0.09	0.24	-0.08	-0.38	-0.23	-0.44	-0.57	-0.56	-0.64	-0.61	-0.60

`defender_deaths_std, standard`

Lots there. Lets talk about the very last table, so the one right above this text. This shows the difference in the variance of the attacker winning when you go from standard rules to table rules. My takeaways are:

1. Slightly lower variance in larger battles when the attacker is outnumbered by a a little (middle/bottom triangle)
2. Much higher variance when the defender is outnumbered (top/right)
3. Very little change when the battle is even
4. Higher variance when there are fewer attackers (left side)

The other one to talk about is the percentage chance that the attacker wins, which is the fourth plot from the bottom. This shows that the table rules are just a straight debuff for the attacker, which gets worse the more defenders there are.

```
In [ ]:
```