Solution to May 15, 2020 Classic Riddler

Gregory Janesch

May 17, 2020

Introduction to the Problem

This problem originates from the May 15, 2020 edition of 538's weekly Riddler. The problem is described thus:

The fifth edition of Dungeons & Dragons introduced a system of "advantage and disadvantage." When you roll a die "with advantage," you roll the die twice and keep the higher result. Rolling "with disadvantage" is similar, except you keep the lower result instead. The rules further specify that when a player rolls with both advantage and disadvantage, they cancel out, and the player rolls a single die. Yawn!

There are two other, more mathematically interesting ways that advantage and disadvantage could be combined. First, you could have "advantage of disadvantage," meaning you roll twice with disadvantage and then keep the higher result. Or, you could have "disadvantage of advantage," meaning you roll twice with advantage and then keep the lower result. With a fair 20-sided die, which situation produces the highest expected roll: advantage of disadvantage, disadvantage of advantage or rolling a single die?

Extra Credit: Instead of maximizing your expected roll, suppose you need to roll N or better with your 20-sided die. For each value of N, is it better to use advantage of disadvantage, disadvantage of advantage or rolling a single die?

Simulation, Main Problem

Solving this with a brute-force simulation is pretty straightforward. First, we simulate generate all possible results of rolling with advantage and rolling with disadvantage:

```
D = 20

advantage = matrix(data=NA, nrow=D, ncol=D)
disadvantage = matrix(data=NA, nrow=D, ncol=D)

for(i in 1:D){
    for(j in 1:D){
        advantage[i,j] = max(i,j)
        disadvantage[i,j] = min(i,j)
    }
}

advantage_vector = as.numeric(advantage)
disadvantage_vector = as.numeric(disadvantage)
```

Changing the matrices into vectors is important in a bit, but for the moment it lets us get the expected values and the counts for the possible advantage rolls easily:

```
mean(advantage_vector)
```

```
## [1] 13.825
```

```
table(advantage_vector)
```

```
## advantage_vector
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
```

And for the disadvantage rolls:

```
mean(disadvantage_vector)
```

```
## [1] 7.175
```

```
table(disadvantage_vector)
```

```
## disadvantage_vector
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
## 39 37 35 33 31 29 27 25 23 21 19 17 15 13 11  9  7  5  3  1
```

So now we turn to the "advantage of disadvantage" (AoD) and "disadvantage of advantage" (DoA). This is where turning the matrices into vectors becomes useful: Since the probabilities for each advantage or disadvantage value vary, if we try to just take the advantage and disadvantage and work with those for the AoD and DoA, we need to manage the marginal probabilities for each individual roll, and then incorporate those into the data. In this case, we can maintain that information by producing a 400-by-400 matrix, one value along each axis for each cell in the original matrices, and then take tallies from that. The code looks much the same as it does above:

```
adv_of_disadv = matrix(data=NA, nrow=D^2, ncol=D^2)
disadv_of_adv = matrix(data=NA, nrow=D^2, ncol=D^2)

for(i in 1:D^2){
    for(j in 1:D^2){
        adv_of_disadv[i,j] = max(disadvantage_vector[i], disadvantage_vector[j])
        disadv_of_adv[i,j] = min(advantage_vector[i], advantage_vector[j])
    }
}
```

And our expected values are:

```
mean(as.numeric(adv_of_disadv))
```

```
## [1] 9.833338
```

```
mean(as.numeric(disadv_of_adv))
```

```
## [1] 11.16666
```

Since the dice are d20s, a single die has an expected value of 10.5, so of the three cases, the DoA has the best expected value.

Simulation, Extra Credit

So now we move on to whether it's better to use a single die, AoD, or DoA if we're looking for a value of at least N on the roll.

The single die is the easy case: Its values follow a discrete uniform distribution, so there's a 20/20 chance of rolling a 1 or better, 19/20 for at least 2, and so on until 1/20 for 20 or better (and it can't be better than that).

The other two cases are only slightly harder, since we don't have a nice theoretical result (yet). But we do have the probabilities of each event, courtesy of the 400-by-400 tables above. Counting the results and dividing by 160,000 to get probabilities for the individual die rolls:

```
AoD_probs = as.numeric(table(adv_of_disadv)/160000)
AoD_probs
```

```
##  [1] 0.00950625 0.02659375 0.04090625 0.05259375 0.06180625 0.06869375
##  [7] 0.07340625 0.07609375 0.07690625 0.07599375 0.07350625 0.06959375
## [13] 0.06440625 0.05809375 0.05080625 0.04269375 0.03390625 0.02459375
## [19] 0.01490625 0.00499375
```

```
DoA_probs = as.numeric(table(disadv_of_adv)/160000)
DoA_probs
```

```
##  [1] 0.00499375 0.01490625 0.02459375 0.03390625 0.04269375 0.05080625
##  [7] 0.05809375 0.06440625 0.06959375 0.07350625 0.07599375 0.07690625
## [13] 0.07609375 0.07340625 0.06869375 0.06180625 0.05259375 0.04090625
## [19] 0.02659375 0.00950625
```

From there, we just need the cumulative sums from each value to 20, which is easily done with a little vector reversing, and we have our final result:

```
extra_credit = data.frame(MinRoll = 1:20, OneDie = 20:1/20,
                          AoD = rev(cumsum(rev(AoD_probs))),
                          DoA = rev(cumsum(rev(DoA_probs))))
kable(extra_credit, digits = 3)
```

MinRoll	OneDie	AoD	DoA
1	1.00	1.000	1.000
2	0.95	0.990	0.995
3	0.90	0.964	0.980
4	0.85	0.923	0.956
5	0.80	0.870	0.922
6	0.75	0.809	0.879
7	0.70	0.740	0.828
8	0.65	0.666	0.770
9	0.60	0.590	0.706
10	0.55	0.513	0.636
11	0.50	0.438	0.562
12	0.45	0.364	0.487
13	0.40	0.294	0.410
14	0.35	0.230	0.334
15	0.30	0.172	0.260
16	0.25	0.121	0.191
17	0.20	0.078	0.130
18	0.15	0.044	0.077
19	0.10	0.020	0.036
20	0.05	0.005	0.010

$N = 1$ is a trivial case, as you can't roll less than 1, so you'll always get a high enough roll. For 2 to 13, the disadvantage of the advantage proves to have the highest probability of success, while the single die is your best chance for when your minimum roll must be at least 14.

Analytical Solution, First Part

From a more theoretical view, this problem amounts to calculating the distributions for order statistics.

Let $F(x)$ denote the cumulative distribution function (CDF) for a random variable X, and let $F_G(x)$ denote the cumulative distribution function (CDF) for some statistic $G(X)$.

Consider $X_{(j)}$ as the jth order statistic from a sample of n variables - that is, there are $j-1$ values below it and $n-j$ above - then the CDF is

$$F_{X_{(j)}} = \sum_{k=j}^{n} \binom{n}{k} F(x)^k [1-F(x)]^{n-k}$$

Since the individual dice rolls are samples from a discrete uniform distribution with possible values from 1 to 20, when rolling for advantage or disadvantage, $F(x) = x/20$. The case for rolling with advantage is the maximum of two rolls, so it is the second order statistic:

$$F_{ADV}(x) = F_{X_{(2)}}(x) = \binom{2}{2} \left(\frac{x}{20}\right)^2 \left(1-\frac{x}{20}\right)^0 = \frac{x^2}{400}$$

Similarly, a roll for disadvantage results in

$$F_{DIS}(x) = F_{X_{(1)}}(x) = \binom{2}{1} \left(\frac{x}{20}\right)^1 \left(1-\frac{x}{20}\right)^1 + \binom{2}{2} \left(\frac{x}{20}\right)^2 \left(1-\frac{x}{20}\right)^0 = \frac{40x-x^2}{400}$$

Since these are cumulative sums from 1 to $1 \le x \le 20$, the values for a single point are the CDFs at x minus the CDF at $x-1$. So the probability of a specific value x resulting from a roll with advantage is

$$P(X_{ADV} = x) = \frac{x^2}{400} - \frac{(x-1)^2}{400} = \frac{2x-1}{400}$$

This agrees with the distribution of the counts from the simulation aside from the normalizing factor of 400:

```
(2*(1:20) - 1)
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
```

Similarly, for rolling with disadvantage,

$$P(X_{DIS} = x) = \frac{40x-x^2}{400} - \frac{40(x-1-(x-1)^2}{400} = \frac{41-2x}{400}$$

Which again matches the simulation above (not repeated here for brevity).

Now we come to the advantage of disadvantage and the disadvantage of the advantage. For the AoD, we look for the second order statistic for $F_{DIS}(x)$ with $n = 2$:

$$F_{AoD}(x) = \binom{2}{2} \left(\frac{40x-x^2}{400}\right)^2 = \frac{1600x^2 - 80x^3 + x^4}{400^2}$$

And the probability for a specific value is then

$$P(X_{AoD} = x) = \frac{4x^3 - 246x^2 + 3444x - 1681}{400^2}$$

Sure enough, this matches the distribution of values from before:

```
z <- function(x){return((4*x^3 - 246*x^2+3444*x-1681))}
z(1:20)
```

```
##  [1]  1521  4255  6545  8415  9889 10991 11745 12175 12305 12159 11761
## [12] 11135 10305  9295  8129  6831  5425  3935  2385   799
```

```
sum((1:20)*z(1:20)/400^2)  #AoD expected value
```

```
## [1] 9.833338
```

Next, the DoA is the first order statistic for $F_{ADV}(x)$ with $n = 2$:

$$F_{DoA}(x) = \binom{2}{1}\left(\frac{x^2}{400}\right)\left(1 - \frac{x^2}{400}\right) + \binom{2}{2}\left(\frac{x^2}{400}\right)^2 = \frac{800x^2 - x^4}{400^2}$$

Making the probability of a specific value

$$P(X_{DoA} = x) = \frac{-4x^3 + 6x^2 + 1596x - 799}{400^2}$$

Again, this matches our simulation:

```
z2 <- function(x){return((-4*x^3 + 6*x^2 + 1596*x - 799))}
z2(1:20)
```

```
##  [1]   799  2385  3935  5425  6831  8129  9295 10305 11135 11761 12159
## [12] 12305 12175 11745 10991  9889  8415  6545  4255  1521
```

```
sum((1:20)*z2(1:20)/400^2)  #DoA expected value
```

```
## [1] 11.16666
```

So we have established that the results of the first part are backed up by theory as well as simulation.

Analytical Solution, Extra Credit

The CDF deals with the cumulative probability up to a point:

$$F(x) = P(X \leq x)$$

So in order to figure out the probabilities of getting a roll of N or better, we need to compute 1 minus the corresponding CDF at $x = N - 1$. Since we had to get the CDFs for both the AoD and DoA cases in the previous part, this is fairly easy to accomplish:

```
AoD_CDF = function(x){(1600*x^2 - 80*x^3 + x^4)/400^2}
DoA_CDF = function(x){(800*x^2 - x^4)/400^2}
extra_credit2 = data.frame(MinRoll = 1:20, OneDie = 20:1/20,
                           AoD = sapply(1:20, function(x){1-AoD_CDF(x-1)}),
                           DoA = sapply(1:20, function(x){1-DoA_CDF(x-1)}))
kable(extra_credit2, digits=3)
```

MinRoll	OneDie	AoD	DoA
1	1.00	1.000	1.000
2	0.95	0.990	0.995
3	0.90	0.964	0.980
4	0.85	0.923	0.956
5	0.80	0.870	0.922
6	0.75	0.809	0.879
7	0.70	0.740	0.828
8	0.65	0.666	0.770
9	0.60	0.590	0.706

MinRoll	OneDie	AoD	DoA
10	0.55	0.513	0.636
11	0.50	0.438	0.562
12	0.45	0.364	0.487
13	0.40	0.294	0.410
14	0.35	0.230	0.334
15	0.30	0.172	0.260
16	0.25	0.121	0.191
17	0.20	0.078	0.130
18	0.15	0.044	0.077
19	0.10	0.020	0.036
20	0.05	0.005	0.010

Once again, we've matched the simulation result.

Conclusion

This writeup outlined both a brute-force simulation and an analytical solution to the question posed. Both solutions result in the same answer, strongly suggesting that the results are correct.