FiveThirtyEight's October 15, 2021 Riddler

Emma Knight

October 17, 2021

This week's riddler, courtesy of Eric Thompson-Martin, is about gambling on a baseball series:

Question 1. *Over in the National League Championship Series, the Washington Rationals and the St. Louis Ordinals (known as the "Ords" for short) are also evenly matched. Again, both teams are equally likely to win each game of the best-of-seven series.*

You enter a competition in which you must predict the winner of each of the seven games before the series begins. If any or all of the fifth, sixth or seventh game are not played, you are not credited with predicting a winner.

You win the competition if you predict at least two games correctly. If you optimize your strategy for picking winners, what is the probability you will win the competition?

To give a baseline example of a computation, assume you pick the Rationals to win every game. Then you win unless the Ordinals go 4-0 or 4-1. The odds of a clean sweep are $\frac{1}{16}$, and the odds of a 4-1 series are $\frac{4}{32}$, so you win $\frac{13}{16}$ of the time.

I don't have any real insight into how to optimize this, so I just wrote code that went over all 128 possible predictions and saw how likely they were to win. One (of four) winning strategies was to pick the Rationals to win the first four games, and then to predict the Ordinals to win the last three. The other three strategies are to flip your prediction on the last game (if your prediction on the last game is relevant, then the series must be 3-3 and it's a coin toss), to flip every prediction (there is no difference between the Rationals and the Ordinals after all), or to do both. The first strategy wins unless the Ordinals sweep (a$\frac{1}{16}$ chance) or the ordinals go up 3-1 and then lose (a $\frac{4}{128}$ chance), so you win $\frac{29}{32}$ of the time.

Before giving the code, it was sufficiently generic to let me tweak the length of the series and how many games you needed to get right. A couple other observations: if you need to just get one game right in a series of length $2n-1$, then the best you can do is a $1 - \frac{1}{2^{2n-1}}$ and this can easily be verified to be true: you lose only if you get literally every game wrong, and so your strategy loses when the opposite of your predictions ends the series. If you predict so that the opposite of your predictions is a full series, then you only lose $\frac{1}{2^{2n-1}}$ of the time. Another interesting note: if you had to get n games right in a best of $2n-1$ series, then you could do no better than picking the same team for every game (essentially guessing who wins the series). This is again easy to see: if your prediction is wins for one possible series, then it cannot win for the “opposite series” where you swap which team wins and loses each game as there aren’t enough games for you to win on both series; since each of those series are equally likely, then you can do no better than $\frac{1}{2}$ of winning, and predicting the winner has exactly a $\frac{1}{2}$ chance of winning.

And now, the code:

```
##This generates a list of all predictions for a best of
##n series, where you predict a team wins or loses.
def genPredict(n):
    work = [[]]
    out = []
    while(True):
        if (len(work) == 0):
            return(out)
        seq = work.pop()
        if (len(seq) < n):
            seq1 = seq.copy()
            seq1.append('w')
            seq2 = seq.copy()
            seq2.append('l')
            work.append(seq1)
            work.append(seq2)
        if (len(seq) == n):
            out.append(seq)

##This generates a list of all possible results in a best of
##2n-1 series, with the same notation as above.
def genResult(n):
    work = [[]]
    out = []
    while(True):
        if (len(work) == 0):
            return(out)
        seq = work.pop()
        if ((seq.count('w') == n) or (seq.count('l') == n)):
            out.append(seq)
        if ((seq.count('w') < n) and (seq.count('l') < n)):
            seq1 = seq.copy()
            seq1.append('w')
            seq2 = seq.copy()
            seq2.append('l')
            work.append(seq1)
            work.append(seq2)

##This computes how many games a given prediction gets right
##on a given result.  This assumes len(predict) >= len(result).
```

```
def score(result, predict):
    score = 0
    for i in range(len(result)):
        if (result[i] == predict[i]):
            score += 1
    return(score)

##This couptes how likely a prediction is to get at least goal
##games right in a collection of results.  Again, this assumes
##that len(predict) >= len(results[i]) for all i.
def success(results, predict, goal):
    p = 0
    for i in range(len(results)):
        if (score(results[i], predict) >= goal):
            p += 2**((-1)*len(results[i]))
    return(p)

##This finds all the best predictions for a series of length
##2*series - 1 and outputs the odds of winning as well as all
##the best predictions.
def findWinners(series, goal):
    predicts = genPredict(2*series - 1)
    results = genResult(series)
    win = 0
    winners = []
    for i in range(len(predicts)):
        p = success(results, predicts[i], goal)
        if (p == win):
            winners.append(predicts[i])
        if (p > win):
            win = p
            winners = [predicts[i]]
    return([win, winners])

##This is where you tweak inputs and outputs.
print(findWinners(4, 2))
```