The game of 30s: an analysis

Meilof Veeningen <meilof@gmail.com>

September 4, 2008

Abstract

In this article, we discuss the *game of 30s*, a fairly simple game commonly played as a drinking game. We discuss the rules, various goals one may have in the game, and the results of a calculation of the optimal strategies to achieve these goals.

First, the rules of the game are introduced, and some strategies and goals one may have in the game are discussed. Next, a calculation for the so-called second stage of the game is executed, and an implementation for the simulator of the first stage is discussed. Next, we show how to use its output, and analyze some of its results. Finally, we claim that by some formal definition, the game can be called a 'game of chance'.

1 The game

1.1 Origins of the game

The game of thirties was introduced to me in Groningen somewhere, I think, in august 2008. The game seems to mainly spread mouth-to-mouth, and references on the internet are hard to come by.

There is a Wikipedia page on the game[1] that describes a similar, though slightly different, version of the game. The tactical considerations described in this article largely apply to this variation as well.

On that Wikipedia page, the game is also claimed to have its origins in the Antwerp student life, and the "world championship" of the game is claimed to be held in Antwerp every year. Since this is not backed up anywhere else on the internet as far as I can tell, this seems to be a bit of a joke.

1.2 Rules

The idea of the game is pretty simple: one starts with 6 dices which you throw. Now, you have to put aside at least one dice, and throw with the rest. Continue with this until all dice are put aside, and add up all the points of the dice. This way, you get a score between 6 and 36. Let us call this part of the game the *first stage*.

[1] `http://nl.wikipedia.org/wiki/Dertigen`

This game is usually played as a drinking game, in which case there is also a *second stage*. When played this way, if you score below 30, then you are awarded the difference between 30 and your score in points, where points correspond to a fixed amount of alcoholic beverage.

If you throw *above* 30, then let x (between 1 and 6) be the number of points over 30 you scored. Now you throw the dice, putting aside every dice that has value x; continue as long as you throw new x's. Then the number of x's aside $+ x$ is the amount of points awarded to the person next to you.

If you succeed in putting aside all dice, then you have in all collected $7x$ points for the person next to you. But then the situation gets worse, since this previous score is doubled, and one starts the second stage all over, only now with $x + 1$ (unless $x = 6$, in which case you throw 6s again).

1.3 Strategies

While in the second stage, the game is all luck, in the first stage one has a choice of how many dice to put aside. We call this choice a *strategy*. For any given strategy, we will be able to calculate the statistical expected number of points we scored, the chance we end up above 30 points, etcetera.

One simple strategy may be to, for example, only put away sixes and throw again with everything else. It turns out that this isn't even as bad a strategy as you may think: on average, you will get 29,8 points. It is better than, for example, putting away fives and sixes: this way you will end up with 29,64 points on average.

We will make one basic assumption on strategies: that it is always better to put away a high dice than a low dice. For example, if we have a 4 and a 5, we can chooce to put away the 5, or both, but not just the 4. Given the rules of the game, this is a pretty natural assumption.

Now, let's sort the dice we have thrown, and the dice we have already put aside. For example, let 1456|56 denote the situation where we have already put away a 5 and a 6, and have thrown a 1, 4, 5 and 6. Given such a situation, a strategy then gives a number, in this case between 0 and 3, specifying how many dice to thow again with. Because of our assumption, this will always be the lowest dice For example, let s_6 be the sixes strategy, then we have $s_6(1456|56) = 3$: we throw again with the 1, 4 and 5.

Note that for the sixes strategy, only the dice that we have just thrown influence our decision, and the dice that we have already put away, don't matter. Thus, we could just as well say $s_6(1456) = 3$. Strategies with this property will be called *simple strategies*.

One example of a situation where a simple strategy does not suffice, is for example $45|\cdot$. Say we want to make sure we get at least 30 points. Suppose we have 36|5556, making exactly 30 points. Then we will not throw again, so $s(36|5556) = 0$. But now, consider 36|6666. In this case, putting just the 6 away guarantees 30 points so we might just as well throw the 3 again: $s(36|6666) = 1$.

After playing the game a bit, one may come up with the following *heuristic strategy*:

> If, after removing the highest dice, at least all but one of the remaining stones are 5s or 6s, at least put those away. If the lowest stone is a 4 or higher; keep it, if it is a 3 or lower, throw it again.

This is a pretty good strategy, as we will see later. We will also analyze the strategies to put away all sixes and all fives/sixes.

1.4 Goals

The fact that we can have many different strategies is nice, but it automatically raises the question: what is the *best* strategy? The answer of this question actually depends on the goal you have in the game, as the following simple example will show.

Suppose that in the first turn, one throws 455556, totalling exactly 30 points. If you want to maximize the number of points, then a calculation shows that it actually makes sense to only put aside the 6 and throw the rest all over, giving you an expected score of 30.44. However, if you just want to make sure you don't have to drink yourself (ie, get at least 30 points), you will just stop throwing.

We define the following goals one may have in the game:

- Maximize – maximize the number of points you throw
- Minimize loss – minimize the number of points below 30 you get
- 30 or over – maximize the chance of getting 30 points or more
- Optimal – maximize the difference in points between you and your opponent

Later on, we will calculate the strategies to achieve these goals. Obviously, to maximize our number of points, we do not need to know what dice we have already put aside. Thus, the strategy for the maximization goal is a simple strategy. As the $45|\cdot$ example previously given indicates, the loss minimization and 30 or over goals do not give us simple strategies.

The optimal strategy may need some clarification. It is different from the maximization goal because it also considers the effects of the second stage of the game. This is because if you throw 32, then because of the second stage, the expected number of points for your opponent is higher than 2, so scoring 32 instead of 30 should matter more than scoring 30 instead of 28. Because of this, we also do not expect the optimal strategy for this goal to be simple.

It is because of the optimal strategy that we need to consider the effects of the second stage of the game. Before we move on to calculate the strategies to achieve these goals, we thus first calculate the expected results of the second stage.

2 The second stage

In this section our aim is to find the expected number of points scored in the second stage of the game. Just skip down to the table at the bottom of this section if the calculation doesn't interest you.

First we ignore the starting-over aspect and just calculate the expected number of 'good' dice thrown, and the chance of throwing 6 'good' dice.

For the calculations, we used the Maxima computer algebra system. First we define $d(x, n)$ as the chance that if we are in the second stage with x dice in total, we end up with n dice of the right kind. The chance that we throw i dice of the right kind in one time is

$$\left(\frac{1}{6}\right)^i \cdot \left(\frac{5}{6}\right)^{x-i} \cdot \binom{x}{i},$$

so summing over this we get the following general formula:

$$d(x,n) = \sum_{i=1}^{n} \left(\frac{1}{6}\right)^i \cdot \left(\frac{5}{6}\right)^{x-i} \cdot \binom{x}{i} \cdot d(x-i, n-i),$$

which translates, adding the borderline cases, to the following Maxima expression:

```
d(x,n):=if(x=0) then 1 else if n=0 then (5/6)^x else
        sum((1/6)^i*(5/6)^(x-i)*
        binomial(x,i)*d(x-i,n-i),i,1,n);
```

Evaluating this, the number of times we throw 0, 1, etc times the right dice is 33,4%, 16,2%, 17,5%, 15,0%, 10,6%, 5,7% and 1,7%, respectively.

Now, we let $\beta \approx 0,017$ be the chance of ending up with 6 'good' dice:

```
b:d(6,6);
```

Then, we let $\alpha \approx 1,69$ be the expected number of dice thrown, *given that not all dice were good*:

```
a:sum(d(6,i)*i,i,0,5)/(1-b);
```

Next, let $ps(s, n)$ be the number of points already scored if we started throwing ss, and we got to throw all over again n times:

```
ps(s,n):=if n=0 then s else 2*(ps(s,n-1)+6*min(s+n-1,6))
```

Then we define $pe(s, n)$ be the number of points we expect to throw if we started throwing ss, and we got to throw all over again n times:

```
pe(s,n):=ps(s,n)+ac*min(s+n,6)
```

Observing that the chance that we got to throw all over again exactly i times is $(\beta^i - \beta^{i+1})$, we see that the epected value of the second stage is exactly $\sum_{i=0}^{\infty} pe(s,i)$. Luckily this sum converges pretty quickly, so we can provide estimates by just considering the first few terms. This way, we get:

$s=1$	$s=2$	$s=3$	$s=4$	$s=5$	$s=6$
2.957	5.881	8.805	11.729	14.653	17.544

3 Implementation of the simulator

In order to analyze the first stage of the game, I wrote a simulator in C++ (included as 30s.cpp) that, given a strategy, calculates the expected value of the number of points of the first stage, and also makes a distribution of how often each value occurs.

3.1 Implementation

A few words on the implementation of the simulator. The ScoreProbabilityVector for a strategy calculates how often, if a certain strategy is followed, one ends up at a certain total number of points. It does this by storing probabilities as fractions (in prob) of $6^{totallog}$ so that the exact values, rather than floating-point estimates, are used. The class has functionality for adding and multiplying these probability distributions.

The dice class represents one group of at most 6 dice. Of special interest is the operator++, which allows one to iterate over all groups of dice (e.g. we have (1112)++==(1113) and (1456)++==(1466). Dice are always stored in low-to-high order.

The dices class represents all possible combinations of a certain number of dice, each with their relative probabilities of being thrown (e.g. 66 is thrown 1/36 of times, while 46 has probability 2/36).

Analogous to ScoreProbabilityVector, evalue stores an expected value (such as the expected number of points above 30) as a fraction. It has much the same operations as the ScoreProbabilityVector.

Now, the strategy class is the base class for playing strategies.

Tiem strategy_simple class is for simple strategies. For each combination of dice on the table, it calculates the expected result for the strategy. Strategies only need to implement the following function:

```
virtual int getcap(dice &d) = 0;
```

One implementation is strategy_simple_maximize, which, given a goal, calculates the best strategy to achieve the goal. To implement a goal, the following function of this class needs to be implemented:

```
virtual int simple_valuation(dice& d) = 0;
```

Similarly, strategy_full is for non-simple strategies, and it calculates the expected result for each combination of dice on the table and dice put away. Strategies need to implement

```
virtual int getcap(dice &d1, dice &d2) = 0;
```

and analogously we have strategy_full_maximize for which goals need the implement

```
virtual int valuation(dice &d) = 0;
```

The calculation for full strategies is done in such a way that if two valuations of a set of dice are the same, then the set of dice with highest sum is favoured.

3.2 Implementation of strategies

As an example of a simple strategy, consider the strategy that just keeps fives and sixes:

```
class strategy_simple_sixfive : public strategy_simple {
  public:
        int getcap(dice& d) {
                int cap2 = d.length - 1;
                while (cap2 > 0 && d.values[cap2-1] >= 4) cap2--;
                return cap2;
        }
};
```

Dice with values 5 and 6 have d.values[i]>=4, so in this case we decrease the cutoff index.

3.3 Implementation of goals

To maximize our score, we can define the following simple goal:

```
class strategy_simple_maximize_sum : public strategy_simple_maximize {
  public:
       int simple_valuation(dice& d) { return d.sum(); }
};
```

To maximize our chances of getting above 30, we define the following goal:

```
class strategy_full_maximize_30over : public strategy_full_maximize {
        int valuation(dice& d) {
                if (d.sum() >= 30) return 1; else return 0;
        }
};
```

4 Analyzing the output of the simulator

Running 30s with no arguments gives information on its usage.

Running 30s with the –basic switch prints out the score distributions and the expected value. For example, for the maximize strategy (which maximizes the score), we get:

```
[..]
es[6 dice]: 30.151976
dist: [21] 0 0 0 0 0 0 1 120 5559 140415 2295917 26961804
243186064 1761009258 10548865640 53382512169 231819153536
875005590819 2901709118414 8529157971990 22389985444124
52878106491024 113138747356859 220611662263293 394418890670077
650852554950924 999741100146929 1434473497468812 1934059403898576
2463943616644536 2861800487411056 3042036163586292 2906925982320528
2378161702400352 1546022052862848 709229931628032 193663098021888
{21936950640377856}
```

This means that the expected number of points with this strategy is 30,15, and that in one of every 219..56 times we expect to end up with a score of 6, 120 in every 219..56 times we end up with 7 points, etcetera.

Running the program with the –verbose switch gives the same information, but now also with lines such as:

```
3 5 5 5 6 6  es: 30.843641 cap: 4
```

This means that if the dice 355566 are still on the table, we should throw again with 4 dice, and then we can expect to end up, on average, with 30,84 points. Since the maximization strategy is simple, we do not consider what's on the table.

For non-simple strategies, by contrast, we get advice for every combination of dice thrown and dice put away, and a summary for each combination of dice put away. For example, consider this partial output of the loss minimization strategy:

```
5 6 6 6  | 5 5          es: 33.000000 cap: 0
6 6 6 6  | 5 5          es: 34.000000 cap: 0
5 5 expected profit[4 dice, 5 5 ]: 28.785912
dist: [10] 0 0 0 0 1 26 286 1921 9275 34950 107257 274689 598802
1130739 1907953 2928270 4252198 5887777 6962712 8665736 11827416
8230836 4981764 2132064 531504 0 0 0 0 0 0 0 0 0 0 0 {60466176}
```

This indicates that if we have put away two fives, if we follow this strategy we can expect to end up with 29.79 points on average.

Using the –compare switch, finally, we can compare two strategies. For example, the following is a part of the comparison output between the loss minimization and the 30 or over strategies:

```
5 5 6 6 6 | 1   0       2
1 1 5 6 6 6 |   2       3
```

So if have put away a one and throw 55666, then if we want to minimize our loss, we should stop, but if we want to end up above 30 points, we need to throw the two 5s again. If we throw 115666 then we need to throw the 5 again only if we want to end up above 30.

5 Results

5.1 The optimal strategy

Let us recall the heuristic strategy we defined earlier:

> If, after removing the highest dice, at least all but one of the remaining stones are 5s or 6s, at least put those away. If the lowest stone is a 4 or higher; keep it, if it is a 3 or lower, throw it again.

If there are four dice or less on the table, this rule always gives you the optimal strategy; if there are five or six dice, there are some exceptions. For five dice, these are the exceptions:

```
throw           optimal heuristic
1 5 5 5 5       4       1
1 5 5 5 6       4       1
2 5 5 5 5       4       1
2 5 5 5 6       4       1
3 5 5 5 5       4       1
3 5 5 5 6       4       1
```

So if we have three fives or more and a one, two or three, then we should throw them all again to play optimally.

In the case of throwing with 6 dice, we have the following list of differences:

```
throw           optimal heuristic
1 5 5 5 5 5     5       1
1 5 5 5 5 6     5       1
1 5 5 5 6 6     4       1
2 5 5 5 5 5     5       1
2 5 5 5 5 6     5       1
2 5 5 5 6 6     4       1
3 5 5 5 5 5     5       1
3 5 5 5 5 6     5       1
3 5 5 5 6 6     4       1
4 5 5 5 5 5     5       0
4 5 5 5 5 6     5       0
```

Here the image is slightly more diffuse; for the two lowest combinations of 5s and 6s, it makes sense to start all over again even with a 4 as lowest throw; in the other situation, only with a 1, 2 or 3 this is helpful.

The following table summarizes, for the optimal strategy, the expected number of points for the various numbers of dice:

1	2	3	4	5	6
3,5	8,24	13,42	18,84	24,44	30,15

This table indicates that if we play the game optimally, then we can expect to get 30,15 points.

5.2 Other strategies

We compared the optimal strategies for getting 30 points or more, minimizing our loss, maximizing the difference in points ("aggressive") and getting the maximum number of points, with the heuristic strategy defined earlier. The results are summarized here (we have shown the expected number of points, chance of getting 30 or more, expected points for us, expected points above 30, and expected points for opponenent):

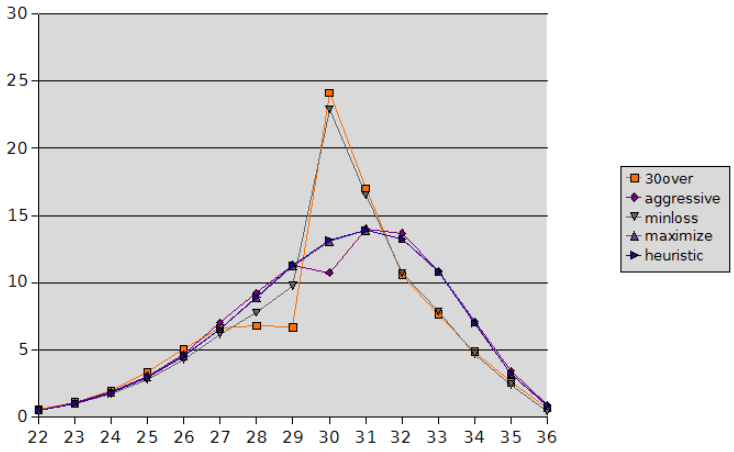
	$E(pnt)$	$P(\geq 30)$	$E(P \leq 30)$	$E(P \geq 30)$	$E(teg)$
30over	29,92	67,44	1,05	0,97	2,86
aggressive	30,13	60,79	1,12	1,25	3,68
minloss	29,95	65,60	1,00	0,95	2,80
maximize	30,15	62,17	1,07	1,23	3,60
heuristic	30,15	62,08	1,07	1,22	3,57

This table provides an indication of what we can achieve in this game within the borders of chance. If we want to drink as little as possible, we can end up with 30 points or more 67,44% of the time, or get on average 1,00 points each time we play.

If we just want to maxmimize our number of points, then we end up above 30 only 62% of times, but our expected number of points, with 1,07, is hardly higher. If we play optimally also considering the second stage of the game, then our expected number of points is raised to 1,12, but the expected number of points for the opponent is raised to 3,68 as opposed to only 2,80 if we want to minimize our own losses.

Finally, the heuristic strategy is somewhat easier to learn than the optimal strategy (an important consideration in drinking games). The expected number of points is the same within two 2 decimals, but the strategy both increases the number of times we end up under 30, and decreases the number of points for the opponent.

The figure on the following page shows the distribution of the expected number of points for the strategies in the table.



6 A game of luck or a game of chance?

As we have seen, different strategies in the game of thirties give us different expected results, even if the game is largely determined by luck. This raises the question: to what extent is the game a game of chance?

Obviously, to answer this question one needs to have a rigurous definition of what exactly is a game of chance. Some research in this area, particularly to be able to advise courts in The Netherlands, has been done by Dr. Ben van der Genugten. The method is explained in this article[2] in the Dutch state newspaper:

The idea is that one defines the learning effect LE in a game to be the expected value of the optimal strategy minus the expected value of a strategy that may be employed by a "beginning player", and the random effect RE to be the expected value of a player with advance knowledge (i.e., he knows what he is going to throw, should he throw again) minis the expected value of a player employing the optimal strategy. Then, the relative skill of a game is defined

$$S = \frac{LE}{LE + RE}.$$

On the grounds of Dutch jurisprudence, the boundary value has been set at 0.20; for example, roulette has value 0, black jack has value 0.049, and some simple forms of poker (the full game is too difficult to analyze) have values around 0.40.

To be able to calculate the random effect, I wrote a small simulator that, for random dice throws, calculates the optimal strategy. This suggests that with prior knowledge, the game has an expected value of about 32.73. As we know, the expected value given an optimal strategy is 30.1520.

So what, in this case, is the relative skill? For this, we need to define what a beginning player would do. Experience suggests that the strategy "put all fives

[2] `http://www.ellylammers.nl/interviewkansspelen.pdf`

and sixes away, and the last dice if it is 4 or higher" may be reasonable. In this case, one gets expected value 29.7232. This would suggest that the relative skill is 0.14, so the game just about a game of chance. This would seem to be reasonable.

There are some pitfalls, though. First of all: is this choice of strategy reasonable? As we know, the heuristic strategy is pretty easy to remember, and has an expected value which is just about equal to the expected value of the optimal strategy. In this case, the relative skill is only 0.0024.

Also, it is a bit difficult to precisely specify a random execution of the game, since the number of different throws may vary. In an earlier version of the simulation, I actually did this wrong. Calculating exactly (i.e., not simulating) the complete expected value with prior knowledge would probably be a very large computation since one needs to consider all possible combinations of putting dice away and leaving them.

7 Conclusion

The game of 30s consists of two stages: the first stage gives you a score, and depending on its score, you may execute the second stage to give points to opponents. In the first stage we have some choice what to do; we call this decision a strategy. We distinguish between simple strategies (that do not depend on the history) and non-simple strategies. One may also have different goals: one may want to avoid getting points, or one may want to give points to opponents.

In the second stage the number of points for the opponent is determined. This involves only luck. We did a calculation in Maxima to determine the expected number of points for the opponent.

In C++, we implemented a simulator that can calculate the expected results for various strategies, and calculate the optimal strategy for a given goal. This simulator can also be used to compare strategies.

If we play the game to maximize our number of points, we can expect to end up with 30,15 points on average. In 62% of all cases we end up with 30 points or more, and our opponent get 3,60 points on average. If we have other goals, we can somewhat stretch these numbers. The heuristic strategy is fairly good substitute for the optimal strategy.

The game of 30s can reasonably be called a game of chance by some formal definition. To what extent this is true depends on what exactly is a 'beginner' in the game, but either way chance plays an imporant role.