A necessary but not sufficient condition for a game being one of skill rather than pure chance is that the player gets to make *choices*. If the game play is fully automatic, as in standard Snakes and Ladders, then there cannot possibly be any skill involved.

However, any game of pure chance can be converted to one of skill simply by adding a wager, or similar tool that sets up decisions for players such as the doubling cube in backgammon (backgammon of course is not a game of pure chance even just in the checker play, but it’s the doubling cube that brings in most of the skill). This is one of the reasons why so much effort over the centuries has gone into understand the probabilities of elementary card and dice games; leveraging the probabilities into a gamble turn a game of low or zero skill into something much more interesting.

**A simple game**

Imagine a simple dice game familiar from “intro to probability” classes, where players A and B take turns, with A starting, and the first to roll a six wins. No choices are involved.

Computing the probability of A winning is a classic exercise in probability pedagogy with an elegant derivation. At the beginning of the game, A obviously has a 1/6 probability of winning straight away, and a 5/6 probability of being disappointed for now and giving the dice to B. At that point, B has 1/6 probability of winning straight away, and a 5/6 probability of having to give the dice back to A… So if p is the probability of winning given you are holding the dice, simply solve:

And some highschool algebra gives the answer as $$p = \frac{6}{11}$$.

So if Player A can convince B to have an even odds bet (“let’s both put in a dollar and whoever wins gets it all”), and to keep playing the game all night (with A always being allowed to start), they’ll come out on average about 9c better for each round they’ve played.

The converse of “no choices means no skill” doesn’t hold true; just having choices doesn’t mean there is skill involved. Consider if, perhaps as part of distracting B from the scam, A introduces a variant. Before each roll, the player has to decide whether to use a red or a blue die. We have choice, but no impact on the game; it remains a game of pure chance. Doubtless silly superstitions (“red always works for me after midnight!”) and rituals (“baby needs new shoes!”) will evolve, and possibly links to political preferences, but the maths of the game doesn’t change.

My hunch is that for psychological reasons players will focus on the things they can control unless they have discipline of iron. I imagine this has been researched, but looking into that will be for another day.

**Simulation**

Games are rarely as simple as the example above. Game designers and players have a knack of introducing complications – I believe with good reason – which mean that direct calculations are often not possible. For instance, I am working on a post on Snakes and Ladders (that’s “Chutes and Ladders” for some, due to an amusing historical concern that snakes on the board were too frightening for American children ), which shows that while it is all very elegant to model it with an absorptive Markov chain as per the research literature, this can only be practically done with simplified rules. Introduce the “3 sixes in a row and its back to square one” rule and things get much, much more complicated.

For even modestly complicated games, simulations will always trump analytical (ie pen and paper) solutions. Before we complicate our dice game, lets simulate the simple version to see if we can replicate the analytical result.

There are plenty of ways to do this, but an efficient way with this super-simple game is to generate a large collection of dice rolls all at once, mark the wins and work out who won by the game length (odd number of rolls means A won). This gives us, in addition to the correct average result, a nice visualisation of the geometric distribution of game lengths:

Here’s the R code for that.

library(tidyverse)

library(scales)

n <- 1000000

dice <- sample(1:6, n, replace = TRUE)

wins <- which(dice == 6)

results <- data.frame(game\_length = diff(c(0, wins))) %>%

mutate(who\_won = ifelse(game\_length %% 2 == 1, "Player A", "Player B"))

rs <- results %>%

group\_by(who\_won) %>%

summarise(freq = n()) %>%

ungroup() %>%

mutate(prop = freq / sum(freq))

ggplot(results, aes(x = game\_length, fill = who\_won)) +

geom\_histogram(binwidth = 1) +

ggtitle("Results of an alternating dice roll game",

paste0("First to roll a six wins; Starting player wins ",

round(rs[1, "prop"], 2), " of the time")) +

scale\_y\_continuous(label = comma) +

labs(x = "Game length", fill = "Winner:",

y = paste("Number of wins out of", format(n, big.mark = ",", scientific = FALSE)))

**A more complex game**

Let’s complicate our game by saying that not only does throwing a six win, so also does getting the same throw as your opponent previously.

This will swing the odds towards Player B, because while A has only a 1/6 chance of winning on their first roll, when Player B gets *their* first turn they have a 1/3 chance (they could roll a six to win, but also win with whatever A just rolled).

This situation is still simple enough to model analytically, but not (for me) in my head, so we’re getting towards territory where a simulation wins out. Actually, I did work this one out analytically to make sure I got the simulation right, so let’s think that through first. We can calculate the probability of the current holder of the dice winning, from the second roll onwards,. This is the same  
method as for the simpler game but with a 1/3 chance of winning on this role rather than 1/6.

The same maths as before gives us a 0.6 probability of winning if you hold the dice, from the second roll onwards. Then we go back to the situation at roll one for Player A and we have:

which yields exactly 0.5 as Player A’s chance at the beginning of the game.

This game is markedly more complicated to simulate than the first one. I tried to use the same approach of generating a big vector of dice roles, then using the tidyverse and extensive use of lag() to implement the game logic, but it was simply too awkward. The complications were issues such as when the dice roll is the same 3 or 4 rolls in a row, identifying in a data\_frame with dice roll as a column exactly which rows marked a win and which ones needed to start the game again. I thought I’d cracked it several times only to discover bugs in rare situations (eg 5 identical non-six numbers in a row). That’s not good, for such a simple game. So I ended up with a completely different and conceptually simpler approach, closer to how humans play the game, of writing a function to play a round of the game. Some people (not me) would regard this as less R-idiomatic because it doesn’t vectorize the whole thing, but it’s much more generalizable to more complex games which is where I’ll be heading in future posts.

*(Trust me, this is going somewhere – by the end of this series I have R successfully playing a 1970s text based computer game with the help of machine learning.)*

That gives us this nice result:

#' Roll a dice and return the game 2 results for one round

#'

#' @param last\_roll the previous roll of the dice. If NA, this means we are at the beginning of the game

#' @return a list with two elements: whether it was a win based on the rule of 6, or matching the last roll;

#' and what the latest roll of the dice is

diceroll <- function(last\_roll){

this\_roll <- sample(1:6, 1)

win <- this\_roll %in% c(6, last\_roll)

return(list(

win = win,

this\_roll = this\_roll)

)

}

#' Main cycle for playing "Game 2"

#'

#' @return the number of rolls it took to win the game

dicegame <- function(){

i <- 0

rolls <- NA

win <- FALSE

while(!win){

i <- i + 1

dr <- diceroll(rolls[length(rolls)])

win <- dr$win

rolls <- c(rolls, dr$this\_roll)

}

return(i)

}

game\_length <- sapply(1:n, function(x){dicegame()})

results <- data.frame(game\_length = game\_length) %>%

mutate(who\_won = ifelse(game\_length %% 2 == 1, "Player A", "Player B"))

rs <- results %>%

group\_by(who\_won) %>%

summarise(freq = n()) %>%

ungroup() %>%

mutate(prop = freq / sum(freq))

svg("..http://freerangestats.io/img/0137-game2-results.svg", 8, 4)

ggplot(results, aes(x = game\_length, fill = who\_won)) +

geom\_histogram(binwidth = 1) +

ggtitle("Results of an alternating dice roll game",

paste0("First to roll a six or to match the last roll wins; Starting player wins ",

round(rs[1, "prop"], 2), " of the time")) +

scale\_y\_continuous(label = comma) +

labs(x = "Game length", fill = "Winner:",

y = paste("Number of wins out of", format(n, big.mark = ",", scientific = FALSE)))

**Key points**

Some basic points I’ll be drawing on again in future posts:

* games with no choices are games of chance
* games of pure chance do not imply equal chances of winning
* any game of chance can be converted to a complex game of skill by adding gambling
* simulation is useful for understanding gaming probabilities
* simulation of complex games is likely to need to mimic the game process (eg turn by turn) rather than elegant mathematical simplifications – precisely because game designers introduce ad hoc complications to their games.