Topics in
Probability Theory and Stochastic Processes
Steven R. Dunbar

Duration of the Gambler’s Ruin

Rating
Mathematically Mature: may contain mathematics beyond calculus with proofs.
**Section Starter Question**

Consider a gambler who wins or loses a dollar on each turn of a fair game with probabilities $p = 1/2$ and $q = 1/2$ respectively. Let his initial capital be $10. The game continues until the gambler’s capital either reduces to 0 or increases to $20$. What is the length of the shortest possible game the gambler could play? What are the chances of this shortest possible game? What is the length of the second shortest possible game? How would you find the probability of this second shortest possible game occurring?

---

**Key Concepts**

1. The principle of first-step analysis, also known as conditional expectations, provides equations for important properties of coin-flipping games and random walks. The important properties include ruin probabilities and the duration of the game until ruin.

2. Difference equations derived from first-step analysis or conditional expectations provide the way to deduce the expected length of the game in the gambler’s ruin, just as for the probability of ruin or victory.

---

**Vocabulary**

1. *Expectation by conditioning* is the process of deriving a difference equation for the expectation by conditioning the outcome over an exhaustive, mutually exclusive set of events, each of which leads to a
simpler probability calculation, then weighting by the probability of each outcome of the conditioning events.

2. **First Step Analysis** is how J. Michael Steele refers to the simple expectation by conditioning process that we use to analyze the ruin probabilities and expected duration. It is a more specific description for coin-tossing games of the more general technique of expectation by conditioning.

---

**Mathematical Ideas**

**Understanding a Stochastic Process**

Start with a sequence of Bernoulli random variables, $Y_1, Y_2, Y_3, \ldots$ where $Y_i = +1$ with probability $p$ and $Y_i = -1$ with probability $q$. Start with an initial value $T_0$ and set $Y_0 = T_0$ for convenience. We define the sequence of sums $T_n = \sum_{i=0}^{n} Y_i$. The goal is to understand the stochastic process $T_1, T_2, T_3, \ldots$. Look at how many trials the process will experience until it achieves the value 0 or $a$. In symbols, consider $N = \min\{n : T_n = 0, \text{ or } T_n = a\}$. Look at the expected value of the number of trials, $D = \mathbb{E}[N]$. This is a special case of a larger class of probability problems called **first-passage distributions** for **first-passage times**.

The principle of first-step analysis, also known as conditional expectations, provides equations for important properties of coin-flipping games and random walks. The important properties include ruin probabilities and the duration of the game until ruin. Difference equations derived from first-step analysis or conditional expectations provide the way to deduce the expected length of the game in the gambler’s ruin, just as for the probability of ruin or victory. **Expectation by conditioning** is the process of deriving a difference equation for the expectation by conditioning the outcome over an exhaustive, mutually exclusive set of events, each of which leads to a simpler probability calculation, then weighting by the probability of each outcome of
the conditioning events. **First Step Analysis** is how J. Michael Steele refers to the simple expectation by conditioning process that we use to analyze the ruin probabilities and expected duration. It is a more specific description for coin-tossing games of the more general technique of expectation by conditioning.

**Expected length of the game**

Note that the following implicitly assumes that the expected duration of the game is finite. This fact is true, see below for a proof.

**Theorem 1.** The expected duration of the game in the classical ruin problem is

\[
D_{T_0} = \frac{T_0}{q-p} - \frac{a}{q-p} \frac{1-(q/p)^{T_0}}{1-(q/p)^a} \quad \text{for} \quad p \neq q
\]

and

\[
T_0(a - T_0) \quad \text{for} \quad p = 1/2 = q.
\]

**Proof.** If the first trial results in success, the game continues as if the initial position had been \(T_0 + 1\). The conditional expectation of the duration conditioned on success at the first trial is therefore \(D_{T_0+1} + 1\). Likewise if the first trial results in a loss, the duration conditioned on the loss at the first trial is \(D_{T_0-1} + 1\).

This argument shows that the expected duration satisfies the difference equation, obtained by expectation by conditioning

\[
D_{T_0} = pD_{T_0+1} + qD_{T_0-1} + 1
\]

with the boundary conditions

\[
D_0 = 0, \quad D_a = 0.
\]

The appearance of the term 1 makes the difference equation non-homogeneous. Taking a cue from linear algebra, or more specifically the theory of linear non-homogeneous differential equations, we need to find the general solution to the homogeneous equation

\[
D^h_{T_0} = pD^h_{T_0+1} + qD^h_{T_0-1}
\]

and a particular solution to the non-homogeneous equation. We already know the general solution to the homogeneous equation is \(D^h_{T_0} = A + B(q/p)^{T_0}\).
The best way to find the particular solution is inspired guessing, based on good experience. Re-write the non-homogeneous equation for the particular solution as

$$-1 = pD_{T_0+1} - D_{T_0} + qD_{T_0-1}.$$  

The right side is a weighted second difference, a difference equation analog of the second derivative. Functions whose second derivative is a constant are quadratic functions. Therefore, it make sense to try a function of the form $D^p_{T_0} = k + lT_0 + mT_0^2$. The exercises show that the particular solution is actually $D_{T_0} = T_0/(q - p)$ if $p \neq q$.

It follows that the general solution of the duration equation is:

$$D_{T_0} = T_0/(q - p) + A + B(q/p)^{T_0}.$$  

The boundary conditions require that

$$A + B = 0$$

and

$$A + B(q/p)^a + a/(q - p) = 0.$$  

Solving for $A$ and $B$,

$$D_{T_0} = \frac{T_0}{q - p} - \frac{a}{q - p} \frac{1 - (q/p)^{T_0}}{1 - (q/p)^a}.$$  

The calculations are not valid if $p = 1/2 = q$. In this case, the particular solution $T_0/(q - p)$ no longer makes sense for the equation

$$D_{T_0} = \frac{1}{2}D_{T_0+1} + \frac{1}{2}D_{T_0-1} + 1.$$  

The reasoning about the particular solution remains the same however, and the particular solution is $-T_0^2$. It follows that the general solution is of the form $D_{T_0} = -T_0^2 + A + BT_0$. The required solution satisfying the boundary conditions is

$$D_{T_0} = T_0(a - T_0).$$  

□
Corollary 1. Playing until ruin with no upper goal for victory against an infinitely rich adversary, the expected duration of the game until ruin is

\[ \frac{T_0}{q-p} \quad \text{for} \quad p \neq q \]

and

\[ \infty \quad \text{for} \quad p = 1/2 = q. \]

Proof. Pass to the limit \( a \to \infty \) in the preceding formulas. \( \Box \)

Proof that the duration is finite

The following discussion of finiteness of the duration of the game is adapted from \[2\] by J. Michael Steele.

The arguments for the probability of ruin or the duration of the game have a logical gap, assuming that the duration \( D_{T_0} \) of the game is finite. Is it sure that the gambler’s net winnings will eventually reach \( a \) or \( 0 \)? This important fact requires proof.

The proof uses a common argument in probability, an “extreme case argument”. Identify an “extreme” event with a small but positive probability of occurring. A complementary “good” event avoids the extreme event. Therefore the complementary “good” avoidance event must happen with probability not quite 1. The avoidance must happen infinitely many independent times, but the probability of such a run of “good” events must go to zero.

For the gambler’s ruin, the event of interest is the game continuing forever. Consider the extreme event that the gambler wins \( a \) times in a row. If the gambler is not already ruined (at 0), then such a streak of \( a \) wins in a row is guaranteed to boost his fortune above \( a \) and end the game in victory for the gambler. Such a run of luck is unlikely, but it has positive probability, in fact, probability \( P = p^a \). Let \( E_k \) denote the event that the gambler wins on each turn in the time interval \([ka, (k + 1)a - 1]\), so the \( E_k \) are independent events. Hence the complementary events \( E_k^C = \Omega - E_k \) are also independent. Then \( D > na \) at least implies that all of the \( E_k \) with \( 0 \leq k \leq n \) fail to occur. Thus, we find

\[ \mathbb{P}[D_{T_0} > na] \leq \mathbb{P}\left[\bigcap_{k=0}^{n} E_k^C\right] = (1 - P)^n. \]

Note that

\[ \mathbb{P}[D_{T_0} = \infty \mid T_0 = z] \leq \mathbb{P}[D > na \mid T_0 = z] \]
for all \( n \). Hence, \( \mathbb{P}[D_{\infty} = \infty] = 0 \), justifying the earlier assumption that the duration must be finite.

**Illustration 1**

The duration can be considerably longer than naively expected. For instance in a fair game, with two players with $500 each flipping a coin until one is ruined, the average duration of the game is 250,000 trials. If a gambler has only $1 and his adversary $1000, with a fair coin toss, the average duration of the game is 999 trials, although some games will be quite short! Very long games can occur with sufficient probability to give a long average.

**Some Calculations for Illustration**

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( T_0 )</th>
<th>( a )</th>
<th>Probability of Ruin</th>
<th>Expected Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>9</td>
<td>10</td>
<td>0.1000</td>
<td>9</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>90</td>
<td>100</td>
<td>0.1000</td>
<td>900</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>900</td>
<td>1,000</td>
<td>0.1000</td>
<td>90,000</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>950</td>
<td>1,000</td>
<td>0.0500</td>
<td>47,500</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>8,000</td>
<td>10,000</td>
<td>0.2000</td>
<td>16,000,000</td>
</tr>
<tr>
<td>0.45</td>
<td>0.55</td>
<td>9</td>
<td>10</td>
<td>0.2101</td>
<td>11</td>
</tr>
<tr>
<td>0.45</td>
<td>0.55</td>
<td>90</td>
<td>100</td>
<td>0.8656</td>
<td>766</td>
</tr>
<tr>
<td>0.45</td>
<td>0.55</td>
<td>99</td>
<td>100</td>
<td>0.1818</td>
<td>172</td>
</tr>
<tr>
<td>0.4</td>
<td>0.6</td>
<td>90</td>
<td>100</td>
<td>0.9827</td>
<td>441</td>
</tr>
<tr>
<td>0.4</td>
<td>0.6</td>
<td>99</td>
<td>100</td>
<td>0.3333</td>
<td>162</td>
</tr>
</tbody>
</table>

**Application to Walks on a Circle**

Stand on a large clock, say on the 1. Now flip a coin and move ahead one hour if the coin turns up heads, and back one hour otherwise. Keep repeating the process until the walk visits all 12 numbers. How long, on average, will this random walk take? If you generalize to clocks with \( H \) positions (for the number of hours), how does the expected time depend on \( H \)?

Let \( V_k \) be the set of hour marks visited at time \( k \), that is, the collection of all vertices already visited up to this time. Let the random variable \( t(n) = \min\{k : |V_k| = n\} \) for \( n \leq H \) be the first time to visit \( n \) distinct vertices.
Then for example it is always the case that \( t(1) = 0 \), and \( t(2) = 1 \). The goal is to calculate \( \mathbb{E}[t(H)] \).

The key realization is that the relation between \( t(n) \) and \( t(n+1) \) is simple. Consider the situation at time \( t(n) \) for \( n < H \). The walk has visited \( n \) vertices and we are at one extreme of the contiguous set of vertices in \( V_{t(n)} \). Adding labels, without loss of generality, say the walk is at 0, and the walk has visited \( 1, 2, \ldots, n-1 \). The next increment \( t(n+1) - t(n) \) is the first time the walk hits either \(-1\) or \( n \) (Note that in the case of \( n = H \), these are the same vertex, but that is irrelevant). Therefore from the basic fact stated above:

\[
\mathbb{E}[t(n+1) - t(n)] = 1 \cdot n = n.
\]

Adding all the increments, the sum telescopes on the left to give the sum on the right

\[
\mathbb{E}[t(n)] = 1 + \cdots + (n-1) = n(n-1)/2
\]

for \( n \leq H \). In particular, \( \mathbb{E}[t(H)] = H(H-1)/2 \).

**Remark.** This conclusion is intuitively plausible for the following reason. At a given time, the “width” or span of the visited sites should be proportional to the square root of the time. In order to visit the full “width” or span \( H/2 \) of sites on either side of the starting point, it should take about time about \( H^2 \). The example makes it explicit and precise.

The following example appeared as a weekly puzzle at the website FiveThirtyEight, *The Riddler, August 4, 2017*.

A class of 30 children is playing a game where they all stand in a circle along with their teacher. The teacher is holding two things: a coin and a potato. The game progresses like this: The teacher tosses the coin. Whoever holds the potato passes it to the left if the coin comes up heads and to the right if the coin comes up tails. The game ends when every child except one has held the potato, and the one who hasn’t is declared the winner. How do a child’s chances of winning change depending on where they are in the circle? In other words, what is each child’s win probability?

In the previous section on the probability of ruin applied to show that the probability of any particular child winning the game is \( 1/30 \). Now the previous analysis shows that the expected duration of the game, when the potato has been in the hands of the teacher and exactly 29 children, leaving only the winner, is \( 30 \cdot 29/2 = 435 \) steps or passes of the potato.
Sources
This section is adapted from [2] with additional background information from [1]. The example of the walks on a circle is adapted from John Cook, Walks on a Clock with additional comments from Jeff Garrett. The example of the coin and potato game appeared as a weekly puzzle at the website FiveThirtyEight, The Riddler, August 4, 2017.

Algorithms, Scripts, Simulations

Algorithm
The goal is to simulate the duration until ruin or victory as a function of starting value. First set the probability $p$, number of Bernoulli trials $n$, and number of experiments $k$. Set the ruin and victory values $r$ and $v$, also interpreted as the boundaries for the random walk. For each starting value from ruin to victory, fill an $n \times k$ matrix with the Bernoulli random variables. Languages with multi-dimensional arrays keep the data in a three-dimensional array of size $n \times k \times (v-r+1)$. Cumulatively sum the Bernoulli random variables to create the fortune or random walk. For each starting value, for each random walk or fortune path, find the duration until ruin or victory. For each starting value, find the mean of the duration until ruin or victory. Finally, find a least squares polynomial fit for the duration as a function of the starting value.

Geogebra GeoGebra script for duration.

R R script for duration.

```r
p <- 0.5
n <- 300
k <- 200

victory <- 10
# top boundary for random walk
ruin <- 0
# bottom boundary for random walk
```
interval <- victory - ruin + 1

winLose <- 2 * (array(0*(runif(n*k*interval) <= p), dim = c(n,k, interval))) - 1

# 0+ coerces Boolean to numeric
totals <- apply( winLose, 2:3, cumsum)
# the second argument `2:3` means column-wise in each panel
start <- outer( array(1, dim=c(n+1,k)), ruin : victory, "*")

paths <- array(0, dim=c(n+1,k,interval))
paths[2:(n+1), 1:k, 1:interval] <- totals
paths <- paths + start

hitVictory <- apply(paths, 2:3, (function(x) match(victory, x, nomatch=n+2)));
hitRuin <- apply(paths, 2:3, (function(x) match(ruin, x, nomatch=n+2)));

# the second argument `2:3` means column-wise in each panel
# If no ruin or victory on a walk, nomatch=n+2 sets the hitting
time to be two more than the number of steps, one more than
time the column length. Without the nomatch option, get NA which
# works poorly with the comparison hitRuin < hitVictory next.

duration <- pmin(hitVictory, hitRuin) - 1
# Subtract 1 since R arrays are 1-based, so duration is 1 less than index
is.na(duration) = duration > n
# Remove durations greater than length of trials
meanDuration = colMeans( duration, na.rm=TRUE)

startValues <- (ruin:victory);
durationFunction <- lm( meanDuration ~ poly(startValues, 2, raw=TRUE) )
# lm is the R function for linear models, a more general view of
# least squares linear fitting for response ~ terms
plot(startValues, meanDuration, col = "blue");
lines(startValues, predict(durationFunction, data=startValues), col = "red")
cat(sprintf("Duration function is: %f + %f x + %f x^2 \n
coefficients(durationFunction)[1], coefficients(durationFunction)[2],
coefficients(durationFunction)[3] ))

Octave script for ruin probabilities

p = 0.5;
n = 300;
k = 200;

victory = 10;
# top boundary for random walk
ruin = 0;
# bottom boundary for random walk

probRuinBeforeVictory = zeros(1, victory-ruin+1);
for start = ruin:victory
    winLose = 2 * (rand(n,k) <= p) - 1;
    # -1 for Tails, 1 for Heads
    totals = cumsum(winLose);
    # -n..n (every other integer) binomial rv sample
    paths = [zeros(1,k); totals] + start;
    victoryOrRuin = zeros(1,k);
    for j = 1:k
        hitVictory = find(paths(:,j) >= victory);
        hitRuin = find(paths(:,j) <= ruin);
        if ( !rows(hitVictory) && !rows(hitRuin) )
            # no victory, no ruin
            # do nothing
        elseif ( rows(hitVictory) && !rows(hitRuin) )
            # victory, no ruin
            victoryOrRuin(j) = hitVictory(1)-1;
            # subtract 1 since vectors are 1-based
        elseif ( !rows(hitVictory) && rows(hitRuin) )
            # do nothing
# no victory, but hit ruin
victoryOrRuin(j) = -(hitRuin(1)-1);
## subtract 1 since vectors are 1-based
else # ( rows(hitvictory) && rows(hitruin) )
  # victory and ruin
  if ( hitVictory(1) < hitRuin(1) )
    victoryOrRuin(j) = hitVictory(1)-1;
  # code hitting victory
  ## subtract 1 since vectors are 1-based
  else
    victoryOrRuin(j) = -(hitRuin(1)-1);
  # code hitting ruin as negative
  ## subtract 1 since vectors are 1-based
  endif
else
endif
endfor

durationUntilVictoryOrRuin(start + (-ruin+1)) = mean(abs(victoryOrRuin));
endfor

durationFunction = polyfit([ruin:victory], durationUntilVictoryOrRuin, 2);
plot([ruin:victory], durationUntilVictoryOrRuin, '+r');
hold on;
x = linspace(ruin, victory, 101);
fittedDuration = polyval(durationFunction, x);
plot(x, fittedDuration, '-');
hold off;
disp("Duration function is a_2 + a_1 x + a_0 x^2 where:")
disp("a_2") , disp(durationFunction(3)),
disp("a_1") , disp(durationFunction(2)),
disp("a_0") , disp(durationFunction(1))

use PDL::NiceSlice;

$p = 0.5;
$n = 300;
$k = 200;
$victory = 10;
$ruin = 0;
$interval = $victory - $ruin + 1;
$winLose = 2 * ( random( $k, $n, $interval ) <= $p ) - 1;
$totals = ( cumusumover $winLose->xchg( 0, 1 )->transpose;
$start = zeroes( $k, $n + 1, $interval )->zlinvals($ruin, $victory);

$paths = zeroes( $k, $n + 1, $interval );

# use PDL::NiceSlice on next line
# Note the use of the concat operator here.
$paths ( 0 : ( $k - 1 ), 1 : $n, 0 : ( $interval - 1 ) ) .= $totals;

$paths = $paths + $start;
$hitVictory = $paths->setbadif( $paths < $victory );
$hitRuin = $paths->setbadif( $paths > $ruin );

$victoryIndex =
( $hitVictory ( , , : )->xchg( 0, 1 )->minimum_ind )
->inplace->setbadtoval( $n + 1 );
$ruinIndex =
( $hitRuin ( , , : )->xchg( 0, 1 )->maximum_ind )
->inplace->setbadtoval( $n + 1 );

$durationUntilRuinOrVictory =
( $victoryIndex->glue( 2, $ruinIndex )->xchg( 2, 1 )
)->xchg( 0, 1 )
->setvaltobad( $n + 1 )->minimum;
( $mean, $prms, $median, $min, $max, $adev, $rms ) =
statsover($durationUntilRuinOrVictory);

use PDL::Fit::Polynomial;
$x = zeroes($interval)->xlinvals( $ruin, $victory );
( $ruinFunction, $coeffs ) = fitpoly1d $x, $mean, 3;
print "Duration function is: ", $coeffs (0), "+", $coeffs (1), "x+",

import scipy

p = 0.5
n = 300
k = 200
victory = 10;
ruin = 0;
interval = victory - ruin + 1

winLose = 2*( scipy.random.random((n,k,interval)) <= p )
- 1
totals = scipy.cumsum(winLose, axis = 0)

start = scipy.multiply.outer( scipy.ones((n+1,k), dtype= int), scipy.arange(ruin, victory+1, dtype=int))
paths = scipy.zeros((n+1,k,interval), dtype=int)
paths[ 1:n+1, :, :] = totals
paths = paths + start

def match(a,b,nomatch=None):
    return b.index(a) if a in b else nomatch
# arguments: a is a scalar, b is a python list, value of nomatch is scalar
# returns the position of first match of its first argument in its second argument
# but if a is not there, returns the value nomatch
# modeled on the R function "match", but with less generality

hitVictory = scipy.apply_along_axis(lambda x:( match(victory,x.tolist()),nomatch=n+2)), 0, paths)
hitRuin = scipy.apply_along_axis(lambda x:match(ruin,x.
tolist()),nomatch=n+2), 0, paths)
# If no ruin or victory on a walk, nomatch=n+2 sets the hitting
# time to be two more than the number of steps, one more than
# the column length.
durationUntilRuinOrVictory = scipy.minimun(hitVictory, hitRuin)

import numpy.ma
durationMasked = scipy.ma.masked_greater(
    durationUntilRuinOrVictory, n)
meanDuration = scipy.mean(durationUntilRuinOrVictory,
    axis = 0)
durationFunction = scipy.polyfit(scipy.arange(ruin, victory+1, dtype=int), meanDuration, 2)
print "Duration function is: ", durationFunction[2], "+", durationFunction[1], "x+", durationFunction[0], "x^2"
# should return coefficients to (x-ruin)*(victory - x), descending degree order

Python script for random walk on a clock

from random import random
from statistics import mean, stdev

p = 0.5
MAXH = 12+1
N = 100000

def time_to_cover_circle(H):
    not_yet_reached = set(range(H))
    count, pos = 0, 0
    while not_yet_reached:
        pos += 1 if random() > p else -1
        pos %= H
        not_yet_reached.discard(pos)
        if not_yet_reached:
            count += 1
        else:
            return count

for H in range(3, MAXH):
    lengths_walks = []
    for i in range(N):
        lengths_walks.append(time_to_cover_circle(H))

15
Problems to Work for Understanding

1. (a) Using a trial function of the form $D_{T_0} = k + lT_0 + mT_0^2$, show that a particular solution of the non-homogeneous equation

$$D_{T_0} = pD_{T_0 + 1} + qD_{T_0 - 1} + 1$$

is $T_0/(q - p)$.

(b) Using a trial function of the form $D_{T_0} = k + lT_0 + mT_0^2$, show that a particular solution of the non-homogeneous equation

$$D_{T_0} = \frac{1}{2}D_{T_0 + 1} + \frac{1}{2}D_{T_0 - 1} + 1$$

is $-T_0^2$.

2. A gambler starts with $2 and wants to win $2 more to get to a total of $4 before being ruined by losing all his money. He plays a coin-flipping game, with a coin that changes with his fortune.

   (a) If the gambler has $2 he plays with a coin that gives probability $p = 1/2$ of winning a dollar and probability $q = 1/2$ of losing a dollar.

   (b) If the gambler has $3 he plays with a coin that gives probability $p = 1/4$ of winning a dollar and probability $q = 3/4$ of losing a dollar.

   (c) If the gambler has $1 he plays with a coin that gives probability $p = 3/4$ of winning a dollar and probability $q = 1/4$ of losing a dollar.
Use “first step analysis” to write three equations in three unknowns (with two additional boundary conditions) that give the expected duration of the game that the gambler plays. Solve the equations to find the expected duration.

3. A gambler plays a coin flipping game in which the probability of winning on a flip is $p = 0.4$ and the probability of losing on a flip is $q = 1 - p = 0.6$. The gambler wants to reach the victory level of $16$ before being ruined with a fortune of $0$. The gambler starts with $8$, bets $2$ on each flip when the fortune is $6$, $8$, $10$ and bets $4$ when the fortune is $4$ or $12$. Compute the probability of ruin in this game.

4. This problem is adapted from *Stochastic Calculus and Financial Applications* by J. Michael Steele, Springer, New York, 2001, Chapter 1, Section 1.6, page 9. Information on buy-backs is adapted from investorwords.com. This problem suggests how results on biased random walks can be worked into more realistic models.

Consider a naive model for a stock that has a support level of $20$/share because of a corporate buy-back program. (This means the company will buy back stock if shares dip below $20$ per share. In the case of stocks, this reduces the number of shares outstanding, giving each remaining shareholder a larger percentage ownership of the company. This is usually considered a sign that the company’s management is optimistic about the future and believes that the current share price is undervalued. Reasons for buy-backs include putting unused cash to use, raising earnings per share, increasing internal control of the company, and obtaining stock for employee stock option plans or pension plans.) Suppose also that the stock price moves randomly with a downward bias when the price is above $20$, and randomly with an upward bias when the price is below $20$. To make the problem concrete, we let $Y_n$ denote the stock price at time $n$, and we express our stock support hypothesis by the assumptions that

\[
\begin{align*}
\mathbb{P}[Y_{n+1} = 21|Y_n = 20] &= 9/10 \\
\mathbb{P}[Y_{n+1} = 19|Y_n = 20] &= 1/10
\end{align*}
\]

We then reflect the downward bias at price levels above $20$ by requiring
that for $k > 20$:
\[
\mathbb{P} [Y_{n+1} = k + 1 | Y_n = k] = \frac{1}{3} \\
\mathbb{P} [Y_{n+1} = k - 1 | Y_n = k] = \frac{2}{3}.
\]

We then reflect the upward bias at price levels below $20$ by requiring that for $k < 20$:
\[
\mathbb{P} [Y_{n+1} = k + 1 | Y_n = k] = \frac{2}{3} \\
\mathbb{P} [Y_{n+1} = k - 1 | Y_n = k] = \frac{1}{3}.
\]

Using the methods of “single-step analysis” calculate the expected time for the stock to fall from $25$ through the support level all the way down to $18$. (There is no reasonable way to solve this problem using formulas. Instead you will have to go back to basic principles of single-step or first-step analysis to solve the problem.)

5. Several North American professional sports leagues have a “best-of-seven” format for their seasonal championship (the World Series for Major League Baseball, the NBA Finals for the National Basketball Association and the Stanley Cup Finals for the National Hockey League.) A best-of-seven playoff is a sequence of games between two teams in which one team must win four games to win the series. If one team has won four games before all seven games have been played, remaining games are omitted.

(a) Explain why or why not the first-step analysis for the expected duration model for victory-or-ruin is sensible for estimating the expected length of a best-of-seven championship series.

(b) How many games would we expect to be needed to complete a best-of-seven series if each team has a 50 percent chance of winning each individual game? What modeling assumptions are you making?

(c) Using the same assumptions how many games are expected to complete a best-of-seven series if one team has a 60 percent chance of winning each individual game? A 70 per cent chance?

(d) Using the same assumptions, make a graph of the expected number of games as a function of $p$, the probability of one team winning an individual game.
6. Perform some simulations of the coin-flipping game, varying $p$ and the start value. How does the value of $p$ affect the experimental duration of victory and ruin?

7. Modify the simulations by changing the value of $p$ and comparing the experimental results for each starting value to the theoretical duration function.

8. Modify the duration scripts to perform simulations of the duration calculations in the table in the section Some Calculations for Illustration and compare the results.

---

**Reading Suggestion:**

**References**


---

**Outside Readings and Links:**

1. [Virtual Labs in Probability](#) Games of Chance. Scroll down and select the Red and Black Experiment (marked in red in the Applets Section.)
Read the description since the scenario is slightly different but equivalent to the description above.)

2. University of California, San Diego, Department of Mathematics, A.M. Garsia A java applet that simulates how long it takes for a gambler to go broke. You can control how much money you and the casino start with, the house odds, and the maximum number of games. Results are a graph and a summary table. Submitted by Matt Odell, September 8, 2003.

I check all the information on each page for correctness and typographical errors. Nevertheless, some errors may occur and I would be grateful if you would alert me to such errors. I make every reasonable effort to present current and accurate information for public use, however I do not guarantee the accuracy or timeliness of information on this website. Your use of the information from this website is strictly voluntary and at your risk.

I have checked the links to external sites for usefulness. Links to external websites are provided as a convenience. I do not endorse, control, monitor, or guarantee the information contained in any external website. I don't guarantee that the links are active at all times. Use the links here with the same caution as you would all information on the Internet. This website reflects the thoughts, interests and opinions of its author. They do not explicitly represent official positions or policies of my employer.

Information on this website is subject to change without notice.
Steve Dunbar’s Home Page, http://www.math.unl.edu/~sdunbar1
Email to Steve Dunbar, sdunbar1 at unl dot edu
Last modified: Processed from \LaTeX source on September 28, 2017