

Random Walk
Essay for Applied Statistics
Zoltán Sándor Kis (EI07G8)
January 19, 2022

1 Introduction

In general, random walks are paths that are the result of some number of random steps in some pre-defined mathematical space. Examples of such random walks are the Brownian motion from physics (the corresponding mathematical space being \mathbb{R}^3), or, the coin tossing random walk—the main topic of this essay.

2 The ideal coin-tossing game

Let us consider a coin-tossing game, where we associate heads with $+1$ and tails with -1 (the assignation is completely arbitrary and can be inverted). We are now going to toss a coin n times, and write down the result of the i th toss as x_i . We can denote the partial sum up to the k th toss as:

$$y_k = \sum_{i=1}^k x_i. \quad (1)$$

From intuition, and based on the well-known (and rather misunderstood) law of large numbers, one would assume that this sum is oscillating back and forth around 0 and it approaches it as the number of tosses approaches infinity, but the reality is quite different: actually the event that the sum is equal to 0 is an almost impossible event:

$$\lim_{n \rightarrow \infty} P(y_n = 0) = 0. \quad (2)$$

We can introduce another new variable, that describes the (signed) *average* value of the tosses, to which the law of large numbers does apply:

$$z_k = \frac{y_k}{k} \quad (3)$$

$$\lim_{n \rightarrow \infty} P(z_n = 0) = 1. \quad (4)$$

Some interesting properties are described by the so-called arcsine laws, described in Theorems 1 to 3 (for proofs, see [1, Chapter 5.4]). For these, we need to know the arcsine distribution. It has a probability density function and cumulative distribution function as such:

$$f(x) = \frac{1}{\pi\sqrt{x(1-x)}} \text{ for } x \in (0, 1) \quad (5)$$

$$F(x) = \frac{2}{\pi} \arcsin(\sqrt{x}) \text{ for } x \in [0, 1]. \quad (6)$$

For easier visualization, we have the plots of the probability density function and the cumulative distribution function shown in Figures 1a and 1b, respectively.

The motivation behind this description is an R script I have made modifications to, the description and the source of which are in Sections 3 and 5, respectively.

Theorem 1 (First arcsine law). *The fraction of all i values at which $y_i > 0$ follows an arcsine distribution.*

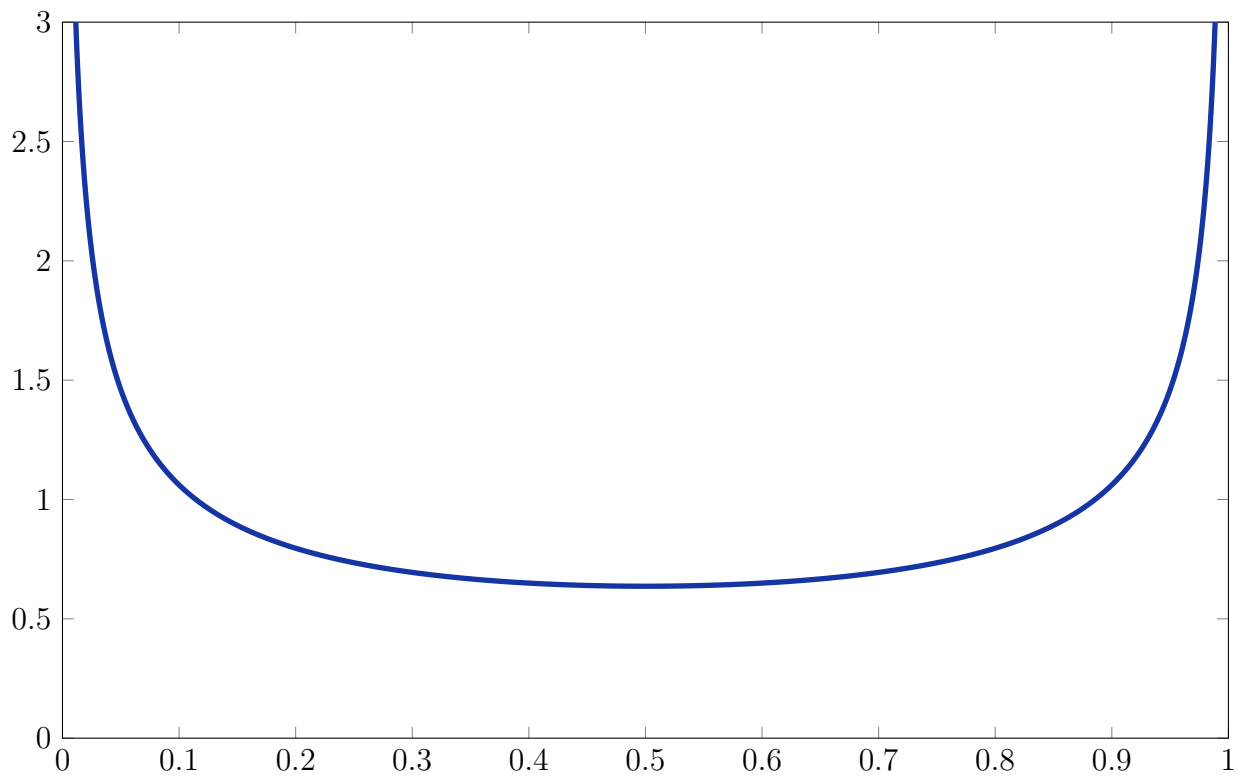
Remark. *The theorem holds true for $y_i < 0$ as well, since the probability of getting heads is the same as getting tails when tossing a fair coin.*

Theorem 2 (Second arcsine law). *The last occasion (i) at which point y_i changes sign (in other words, visits the origin) follows an arcsine distribution, after normalizing with the total length of the run ($x = \frac{i}{n}$).*

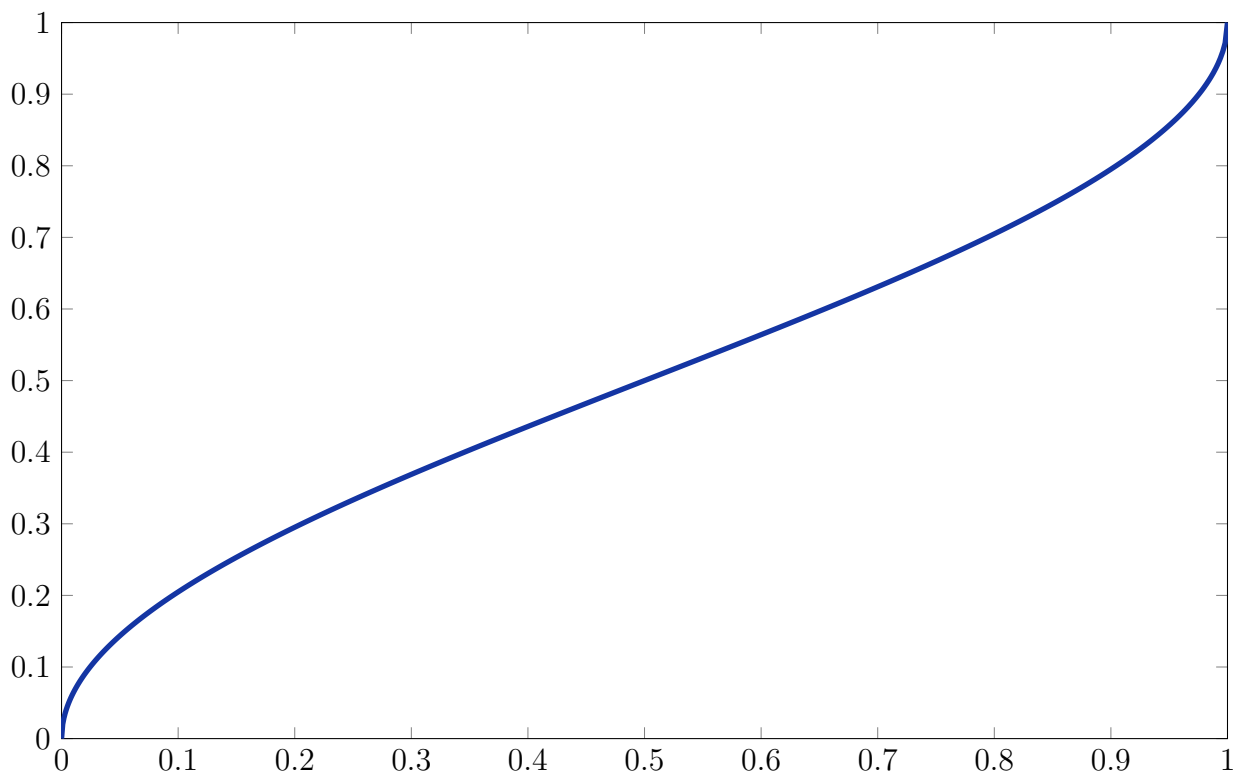
Theorem 3 (Third arcsine law). *The specific i when y_i reaches its maximum in a given run follows an arcsine distribution, after normalizing with the total length of the run ($x = \frac{i}{n}$).*

Remark. *Theorems 2 and 3 are equivalent.*

We shall discuss the seemingly paradoxical nature of these laws further in Sections 2.1 and 2.2.



(a) Arcsine probability density function.



(b) Arcsine cumulative distribution function.

Figure 1. Arcsine distribution.

2.1 Consequences of the first arcsine law

One would assume that roughly half of y_i values of a run—due to equal probability of getting heads and tails—would be positive, and roughly another half negative. As we can see, actually the contrary is the case: the most probable results are that all y_i values are negative and that they are all positive.

2.1.1 Goal paradox

This paradox is essentially a reformulation of the coin-tossing game. Here, we have two teams (let us call them simply A and B) playing some ball game against each other. We assume the two teams' skills are identical, meaning for every goal, the events “team A scored” and “team B scored” are equally probable.

In this scenario, according to Theorem 1, it is the *least* probable that team A and team B are in the lead for the same amount of time. For instance, if there were 20 goals in a game, the probability of one team being in the lead after some 10 of the goals and the other after the other 10 is merely 6%, meanwhile the probability that one team was in the lead for the whole game is 35%. These percentages were calculated using a Python script that takes into account all possible matches. For the source code, see Section 4.

2.2 Consequences of the second arcsine law

As in Section 2.1, we once again arrive at a very unintuitive result. One would expect that since the coin is fair, y_i would oscillate around 0, so the last time it changing sign would be probably rather soon, and would decrease as we move backwards in the sequence of tosses. As we see, our first assumption (the change of sign being probable at the last moments) *is* true, but the latter one is not, since it is equally possible that the last change of sign occurs very close to the start of the run.

A real-world application of this is in the stock market [2]. Let us assume that the stock market acts in an unpredictable, random manner, and the price of an asset can be approximated with a one-dimensional random walk on the field of rational numbers \mathbb{Q} , its steps happening with a constant (and high) frequency. (We have to assume a unit step length as well, thus this approximation is only correct in fairly stable markets.) In this case, in a given day (24 hours)

the probability of the stock price reaching its maximum during the last 4 hours is 27%, about 1.5 times as much as one would expect ($\frac{4}{24} \approx 17\%$). The probability of reaching the maximum in the last hour is 13% instead of the expected 4%. The values of the stock reaching its maximum for 2 hour periods of a day are given in Table 1, which were calculated via an approximation using the arcsine distribution, see Equation (7), where k denotes the upper hour limit of the interval (with the arbitrary starting point defined at 0, e.g. for the interval between 8 and 10 hours after the starting point $k = 10$). (Needless to say, this reasoning applies to the minimum as well, so this is not some “trick” for the stock market.)

$$P(k) = \frac{2}{\pi} \left(\arcsin\left(\sqrt{\frac{k}{24}}\right) - \arcsin\left(\sqrt{\frac{k-2}{24}}\right) \right) \quad (7)$$

Table 1. Probability of a stock reaching its maximum price in a given 2 hour interval in a 24 hour day using the approximation given in Equation (7).

time interval (hours)	P / %
0–2	18.6
2–4	8.1
4–6	6.6
6–8	5.8
8–10	5.5
10–12	5.3
12–14	5.3
14–16	5.5
16–18	5.8
18–20	6.6
20–22	8.1
20–24	18.6

3 Documentation of the R program

This program was the main project along with this essay, mentioned briefly in Section 2. The program simulates several (ideal) coin-toss experiments. An experiment consists of n_{toss} number of tosses, and this experiment is repeated n_{rep} times. At every repetition, it calculates the proportion of time (not in the temporal sense) when the distance was positive compared to the full length (n_{toss}) as $a = \frac{\sum_i^{n_{\text{toss}}} [y_i > 0]}{n_{\text{toss}}}$, where $[.]$ denotes the Iverson bracket, i.e. it is equal to 1 if the statement therein is true and 0 otherwise. The program prints the cumulative distance y_i and its average z_i with respect to the number of tosses i for every $n_{\text{intermed_plots}}$ experiments (an example shown in Figure 2a). After all n_{rep} repetitions are over, it plots a histogram of the proportions of the runs with respect to their a value, and shows the theoretical arcsine probability distribution function above it (an example shown in Figure 2b). An alternative version of the script has also been written, in which the aforementioned positive proportion a is calculated at all i as $a_i = \frac{\sum_j^i [y_j > 0]}{n_{\text{toss}}}$, not only after the end of each run (thus, this has 999 times more data—the leading zeros are not taken into account). The difference in both the script source and the resulting histogram are shown in Section 5.1 and Figure 3, respectively.

The number of tosses in an experiment n_{toss} and the number of repetitions of the experiment n_{rep} are given in lines 22 and 23, as `n_toss` and `n_rep`, respectively.

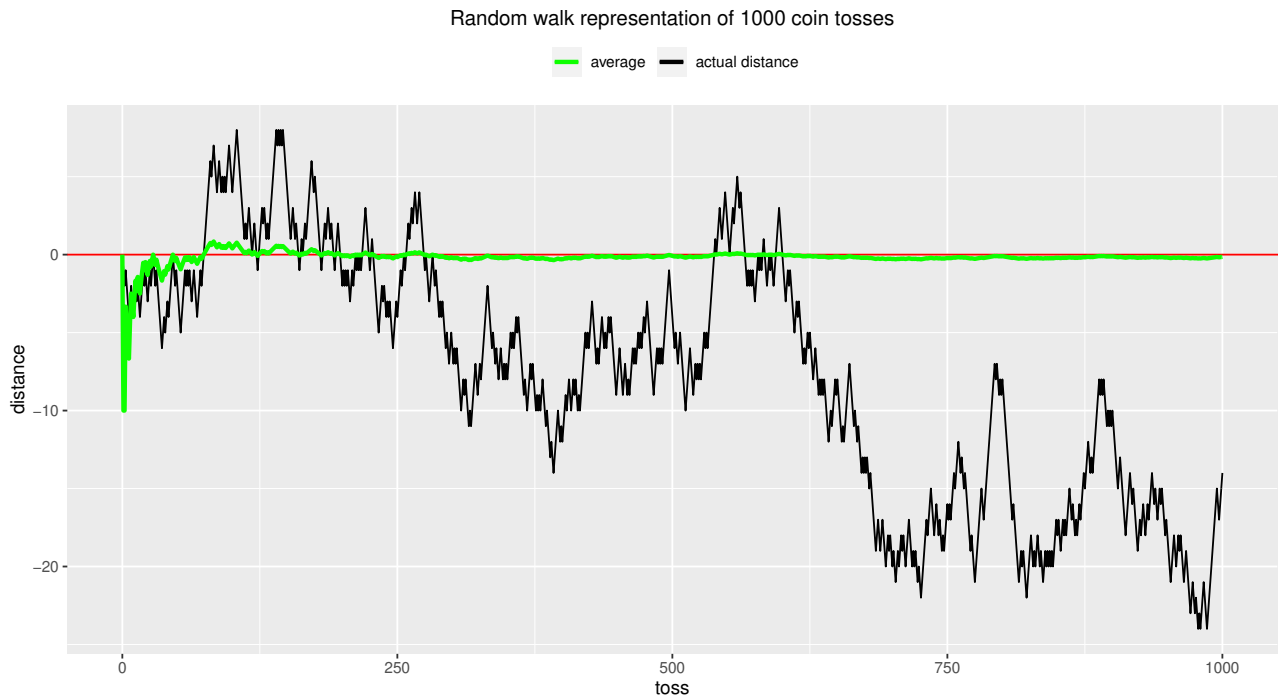
After this, we can give the number of repetitions for which an intermediate result is to be displayed $n_{\text{intermed_plots}}$ as `intermed_plots`. Setting this to 0 will result in no intermediate results displayed. $n_{\text{intermed_plots}} \equiv 0 \pmod{n_{\text{rep}}}$ must hold (i.e. n_{rep} should be an integer multiple of $n_{\text{intermed_plots}}$), else no intermediate results will be displayed.

The user can set if they want the program to pause after every intermediate result by setting `halt` to true or false. If it is true, then `pause_time` will define for how long (in seconds) the script stops before moving on.

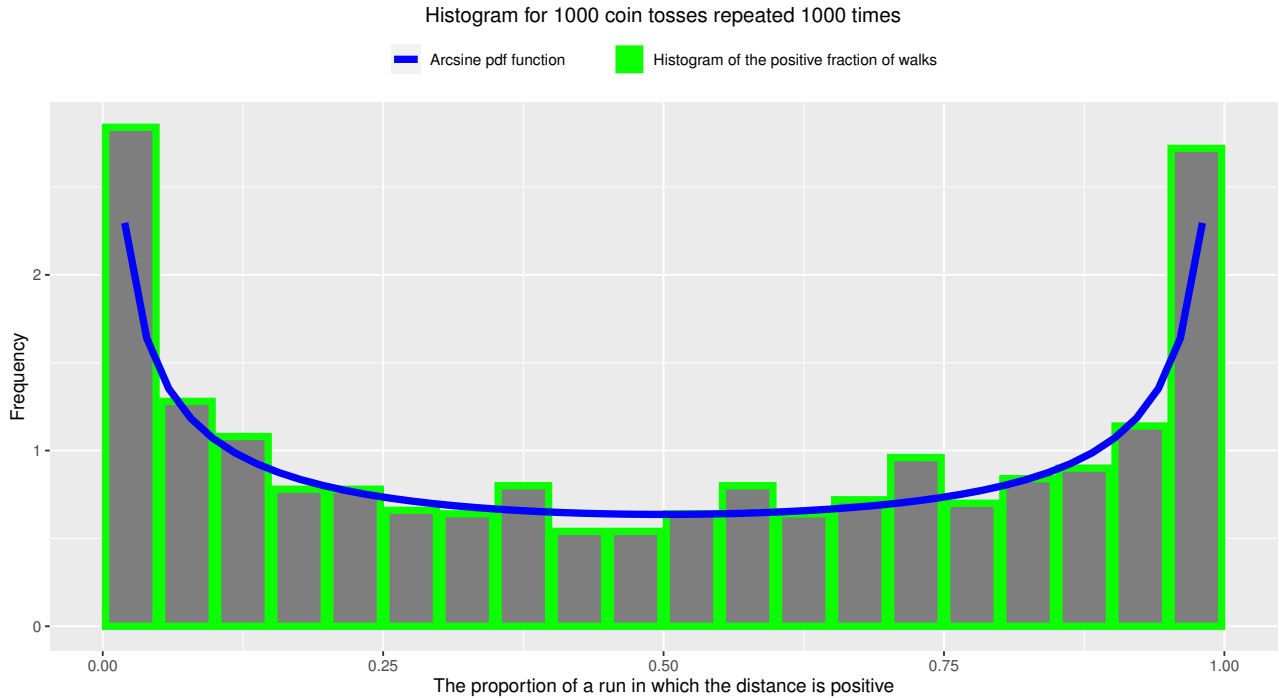
At the end, the program plots the results in a histogram, with the theoretical arcsine probability density function plotted on it. `n_bins` defines how many bins the histogram has, while `n_points_for_arcsin` defines how many point we calculate the arcsine probability density function at.

Finally, `rngseed` defines the seed for the random number generation. Setting the same seed, we can get exactly the same sequential series of tosses, as the pseudo-random number generator

provides always the same series of random numbers starting from the same seed.



(a) An example of an intermediate plot. Distances y_i are displayed as a function of the toss number i .



(b) An example of the histogram showing the relative frequency of runs with respect to the proportion when the distance y_i is in the positive domain.

Figure 2. Examples of resulting plots of the script.

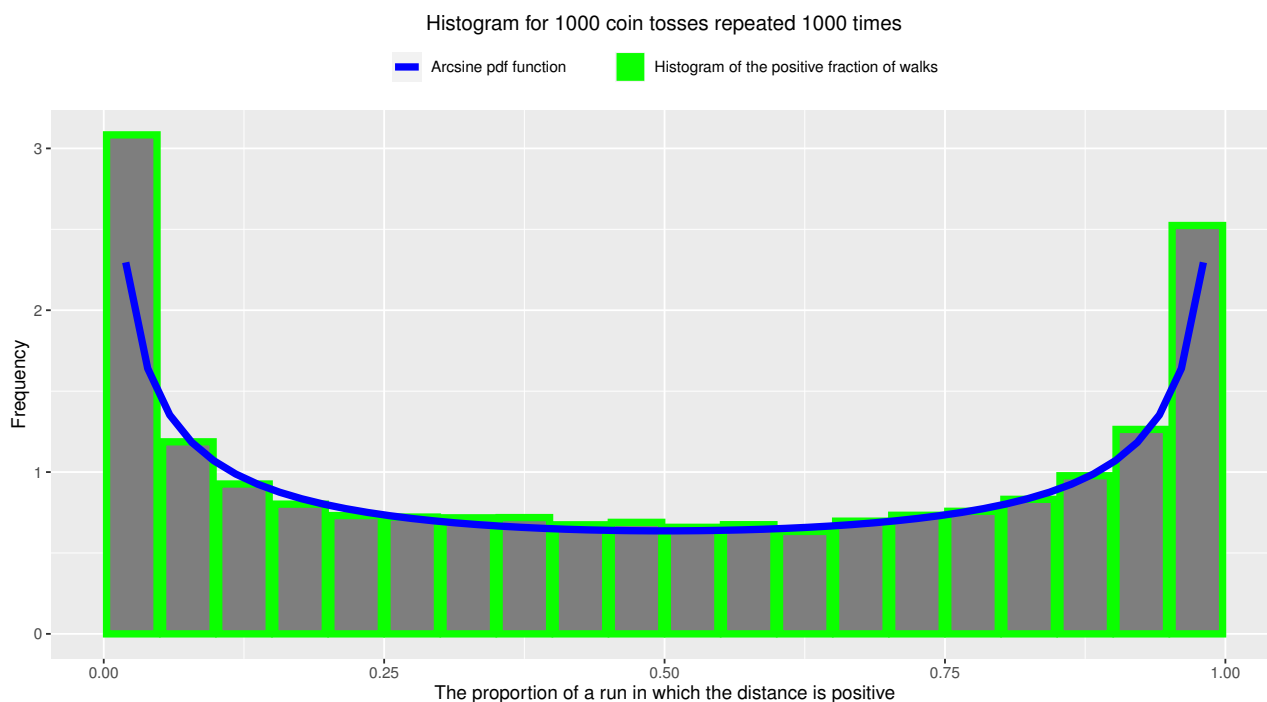


Figure 3. An example of the histogram showing the relative frequency of runs with respect to the proportion when the distance y_i is in the positive domain, calculated at all i .

4 Source code of the Python script for the calculation of goal-paradox values

```
1  import BitVector as bv
2  vectorsize = 20
3  halfLeadCounter = 0
4  allLeadCounter = 0
5
6  for i in range(2**vectorsize):
7      walk = bv.BitVector(intVal=i, size=vectorsize)
8      position = 0
9      positiveCounter = 0
10     negativeCounter = 0
11
12     for j in range(vectorsize):
13         if (walk[j] == 0):
14             position -= 1
15         elif (walk[j] == 1):
16             position += 1
17
18         if (position > 0):
19             positiveCounter += 1
20         elif (position < 0):
21             negativeCounter += 1
22         else:
23             if (walk[j] == 0):
24                 positiveCounter += 1
25             else:
26                 negativeCounter += 1
27
28     if (positiveCounter == negativeCounter):
29         halfLeadCounter += 1
30
31     if (positiveCounter == vectorsize or negativeCounter == vectorsize):
32         allLeadCounter += 1
33
34 print(halfLeadCounter/2**vectorsize)
35 print(allLeadCounter/2**vectorsize)
```

5 Source code of the R program described in Section 3

```
1  # The script uses packages VaRES and ggplot2; thus, these should be  
   ↪ installed prior to running the script.  
2  require(VaRES)  
3  require(ggplot2)  
4  require(profvis)  
5  
6  # Input values to be specified by the user  
7  
8  n_toss <- 1000           # Number of tosses in one experiment  
9  n_rep <- 1000           # Number of repetitions of the experiment  
10 intermed_plots <- 100   # Number of repetitions after which to display  
   ↪ intermediate results (for no intermediate plots: 0)  
11                           # (criterion: intermed_plots = 0 (mod n_rep))  
12 halt <- FALSE           # Boolean that describes whether or not to halt  
   ↪ after every intermediate graph  
13 pause_time <- 3         # Number of seconds of waiting before going on  
   ↪ with the script (this has an effect only if halt == TRUE)  
14 n_points_for_arcsin <- 50 # Number of points on the arcsine pdf function  
15 n_bins <- 20            # Number of bins in the histogram  
16 rngseed <- 1111         # Seed for the random number generation  
17  
18 # User-intervention ends here  
19  
20 set.seed(rngseed)  
21 coll_mat <- matrix(data = NA, nrow = n_rep, ncol = 1)  
22 colnames(coll_mat) <- as.character(n_toss)  
23 for(k in 1:n_rep){  
24   rand_vec <- runif(n_toss, -1, 1)  
25   exp_vec <- character(0)  
26   step_vec <- integer(0)  
27   walk_vec <- 0  
28   avg_vec <- 0  
29   dist_vec <- integer(0)  
30   for(j in 1:length(rand_vec)){  
31     if(rand_vec[j] >= 0){  
32       #exp_vec <- c(exp_vec, "Heads")  
33       step_vec <- c(step_vec, -1)
```

```
34     walk_vec <- c(walk_vec, walk_vec[length(walk_vec)] - 1)
35     avg_vec <- c(avg_vec, walk_vec[length(walk_vec)] / j)
36   } else {
37     #exp_vec <- c(exp_vec, "Tails")
38     step_vec <- c(step_vec, 1)
39     walk_vec <- c(walk_vec, walk_vec[length(walk_vec)] + 1)
40     avg_vec <- c(avg_vec, walk_vec[length(walk_vec)] / j)
41   }
42 }
43
44 # Plotting and graphics unit starts here #
45
46 if(intermed_plots != 0){
47   if((n_rep %% intermed_plots) == 0){
48     if(k%%intermed_plots == 0){
49       print(k)
50       walk_tab <- as.data.frame(cbind(c(0:n_toss), walk_vec, 10*avg_vec))
51       colnames(walk_tab) <- c("toss", "distance", "average")
52       print(ggplot(data = walk_tab, aes(x = toss)) +
53             geom_line(aes(y = distance, colour = "actual distance")) +
54             geom_abline(slope = 0, intercept = 0, col = "red")+
55             geom_line(aes(y = average, colour = "average"), size = 1.2)
56             ↪ +
57             labs(title = paste("Random walk representation of",
58             ↪ as.character(n_toss), "coin tosses", sep = " "), color =
59             ↪ "") +
60             theme(plot.title = element_text(size = 12, hjust = 0.5),
61             ↪ legend.position = "top") +
62             scale_color_manual(values = c("average" = "green", "actual
63             ↪ distance" = "black")))
64     if (halt) {
65       Sys.sleep(pause_time)
66     }
67   }
68 }
69 coll_mat[k,which(colnames(coll_mat) == as.character(n_toss))]<-
70   ↪ length(which(walk_vec > 0)) / length(walk_vec)
71 }
```

```
67   for(i in 1:ncol(coll_mat)){
68     histogram <- hist(coll_mat[,i], plot = F, breaks = n_bins)
69     x_hist <- histogram$mids
70     dens_hist <- histogram$density
71     hist_frame <- as.data.frame(cbind(x_hist, dens_hist))
72     arcsinpoints <- matrix(data = 0, nrow = n_points_for_arcsin, ncol = 1)
73     for (j in 1:n_points_for_arcsin){
74       arcsinpoints[j] <- 0 + (j/(n_points_for_arcsin+1))
75     }
76     darcsinpoints <- darcsine(arcsinpoints)
77     data.frame(lapply(hist_frame, "length<-", max(lengths(hist_frame))))
78     print(ggplot() + geom_col(aes(x = hist_frame$x_hist, y =
      ↪ hist_frame$dens_hist, fill = "Distance histogram"), size = 2, col =
      ↪ "green") +
79       geom_line(aes(x = arcsinpoints, y = darcsinpoints, colour =
      ↪ "Arcsine pdf function"), size = 2) +
80     labs(x = "The proportion of a run in which the distance is
      ↪ positive", y = "Frequency", title = paste("Histogram for",
      ↪ colnames(coll_mat)[i], "coin tosses repeated",
      ↪ as.character(n_rep), "times", sep = " "), color = "", fill =
      ↪ "")+
81     theme(plot.title = element_text(size = 12, hjust = 0.5),
      ↪ legend.position = "top")+
82     scale_color_manual(values = c("Arcsine pdf function" = "blue")) +
83     scale_fill_manual(values = c("Histogram of the positive fraction
      ↪ of walks" = "green"))))
84   }
```

5.1 Alternative histogram script

```
...  
35 coll_mat <- matrix(data = NA, nrow = n_rep, ncol = n_toss-1)  
36 colnames(coll_mat) <- character(length = n_toss-1)  
37 for (i in 1:n_toss-1){  
38   colnames(coll_mat)[i] <- as.character(i+1)  
39 }  
...  
82 for (i in 2:n_toss){  
83   coll_mat[k,which(colnames(coll_mat) == as.character(i))] <-  
    ↪ length(which(walk_vec[2:i] > 0)) / length(walk_vec[2:i])  
84 }  
...  
88 histogram <- hist(coll_mat, plot = F, breaks = n_bins)
```

References

- [1] Peter Mörters and Yuval Peres. *Brownian Motion*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2010. DOI: [10.1017/CB09780511750489](https://doi.org/10.1017/CB09780511750489).
- [2] M. G. Kendall and A. Bradford Hill. “The Analysis of Economic Time-Series-Part I: Prices”. In: *Journal of the Royal Statistical Society. Series A (General)* 116.1 (1953), pp. 11–34. ISSN: 00359238. URL: <http://www.jstor.org/stable/2980947>.