Exercises from Chapter 1 of Introduction to Stochastic Processes with R by Robert P. Dobrow

Ojeda Contreras Braulio Melquisedec

2022-10-15

1.32) Simulate flipping three fair coins and counting the number of heads X.

First, we are going to construct our function to simulate the flipping of n coins with certain probability p of getting head on a flip.

```
flip_coins <- function(n, p) {
  flips <- integer()

  for (i in 1:n) {
    flips[i] <- sample(c(0, 1), 1, prob = c(1 - p, p))
  }

  return (flips)
}
```

With the above function we are ready to solve both subtasks...

(a) Use your simulation to estimate P(X = 1) and E(X).

```
n <- 3
p <- 1/2
trials <- 100000

res <- replicate(trials, flip_coins(n = n, p = p))
x <- apply(res, 2, sum)
prob_x_1 <- length(x[x == 1]) / trials
expectation <- mean(x)

cat("P(X=1)=", prob_x_1, "\tE(X)=", expectation)
```

```
## P(X=1)= 0.37448  E(X)= 1.50294
```

(b) Modify the above to allow for a biased coin where P(Heads) = 3 / 4.

```r
n <- 3
p <- 3/4
trials <- 100000

res <- replicate(trials, flip_coins(n = n, p = p))
x <- apply(res, 2, sum)
prob_x_1 <- length(x[x == 1]) / trials
expectation <- mean(x)

cat("P(X=1)=", prob_x_1, "\tE(X)=", expectation)
```

```
## P(X=1)= 0.14045  E(X)= 2.24996
```

1.33) Cards are drawn from a standard deck, with replacement, until an ace appears. Simulate the mean and variance of the number of cards required.

First, we define our function to simulate the pick of cards with replacement from a standard 52-card deck.

```r
pick_ace_std_deck <- function() {
  # We are going to consider the 4 suits of deck but skipping the jokers.
  # Also we are taking into account the next symbology for each suit:
    # c: clubs
    # d: diamonds
    # h: hearts
    # s: spades
  # and likewise
  # a: ace
  # j: jack
  # q: queen
  # k: king
  deck <- c("a_c", "2_c", "3_c", "4_c", "5_c", "6_c", "7_c", "8_c", "9_c", "10_c",
            "j_c", "q_c", "k_c",
            "a_d", "2_d", "3_d", "4_d", "5_d", "6_d", "7_d", "8_d", "9_d", "10_d",
            "j_d", "q_d", "k_d",
            "a_h", "2_h", "3_h", "4_h", "5_h", "6_h", "7_h", "8_h", "9_h", "10_h",
            "j_h", "q_h", "k_h",
            "a_s", "2_s", "3_s", "4_s", "5_s", "6_s", "7_s", "8_s", "9_s", "10_s",
            "j_s", "q_s", "k_s")
  total_cards <- length(deck)
  picked_cards <- character()
  ace <- FALSE
  i <- 1

  while (ace == FALSE) {
    picked_cards[i] <- sample(deck, 1, prob = rep(c(1 / total_cards), total_cards))
    if (substr(picked_cards[i], 1, 1) == "a")
      ace <- TRUE
    else
      i <- i + 1
  }
  #print(picked_cards)
```

```
  return(i)
}
```

Then, we are going to use the previous function by executing it many times in order to get an precise approximation of the number of cards required to be picked before getting an ace.

```
trials <- 100000
res <- replicate(trials, pick_ace_std_deck())
cards_required_mean <- mean(res)
cards_required_var <- var(res)

cat("Mean=", cards_required_mean, "\tVariance=", cards_required_var)
```

```
## Mean= 12.95398   Variance= 155.2495
```

1.34) The time until a bus arrives has an exponential distribution with mean 30 minutes.

(a) Use the command rexp() to simulate the probability that the bus arrives in the first 20 minutes.

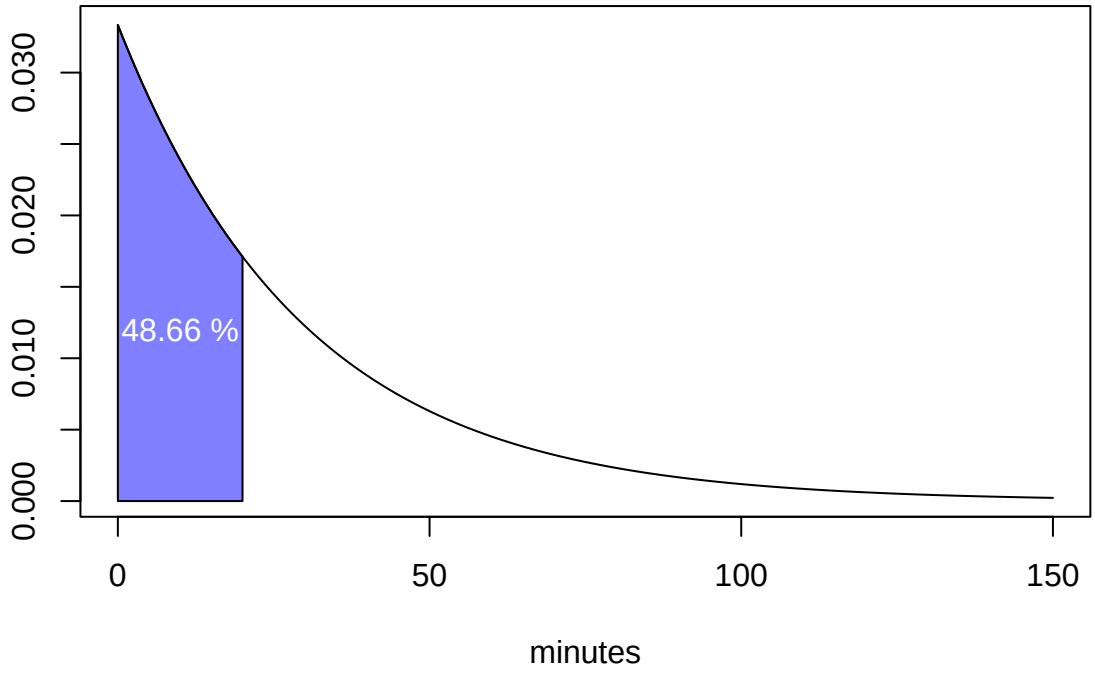
```
n <- 1000000 # number of observations
ari_mean <- 30
rate <- 1 / ari_mean

set.seed(1)
rand_gen_exp <- rexp(n, rate)
prob_x_20 <- length(rand_gen_exp[rand_gen_exp <= 20]) / n

cat("P(X<=20) =", prob_x_20)
```

```
## P(X<=20) = 0.486253
```

(b) Use the command pexp() to compare with the exact probability.

```
exp_area <- function(rate, lb, ub, acolor = "lightgray") {
    x <- seq(0, 150, 0.01)

    if (missing(lb)) {
       lb <- min(x)
    }
    if (missing(ub)) {
        ub <- max(x)
    }

    x2 <- seq(lb, ub, length = 100)
    plot(x, dexp(x, rate = rate), type = "n", xlab = "minutes", ylab = "")
```

```
    y <- dexp(x2, rate = rate)
    polygon(c(lb, x2, ub), c(0, y, 0), col = acolor)
    lines(x, dexp(x, rate = rate), type = "l")
}

# Calculate exact probability
exact_prob_x_20 <- pexp(20, rate = 1 / ari_mean)
percentage <- round(exact_prob_x_20 * 100, 2)

# Plotting the distribution
exp_area(rate, lb = 0, ub = 20, acolor = rgb(0, 0, 1, alpha = 0.5))
text(10, 0.012, paste(as.character(percentage) ,"%"), col = "white", cex = 1)
```



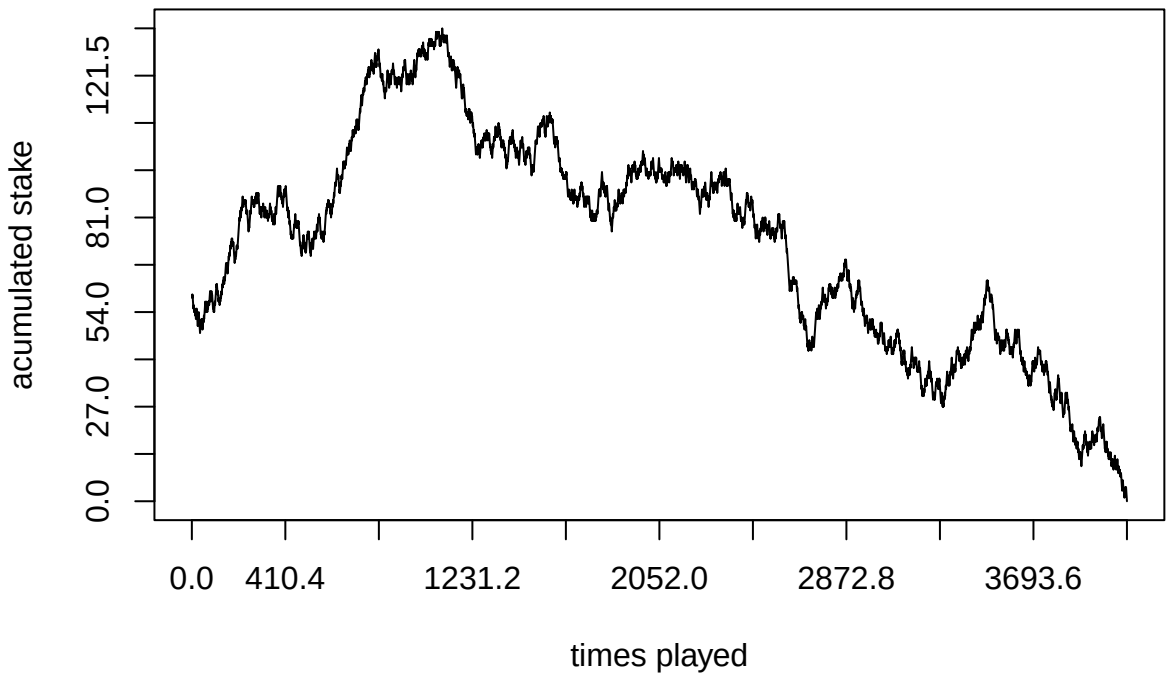
1.35) See the script file gamblersruin.R. A gambler starts with a $60 initial stake.

(a) The gambler wins, and loses, each round with probability p = 0.50. Simulate the probability the gambler wins $100 before he loses everything.

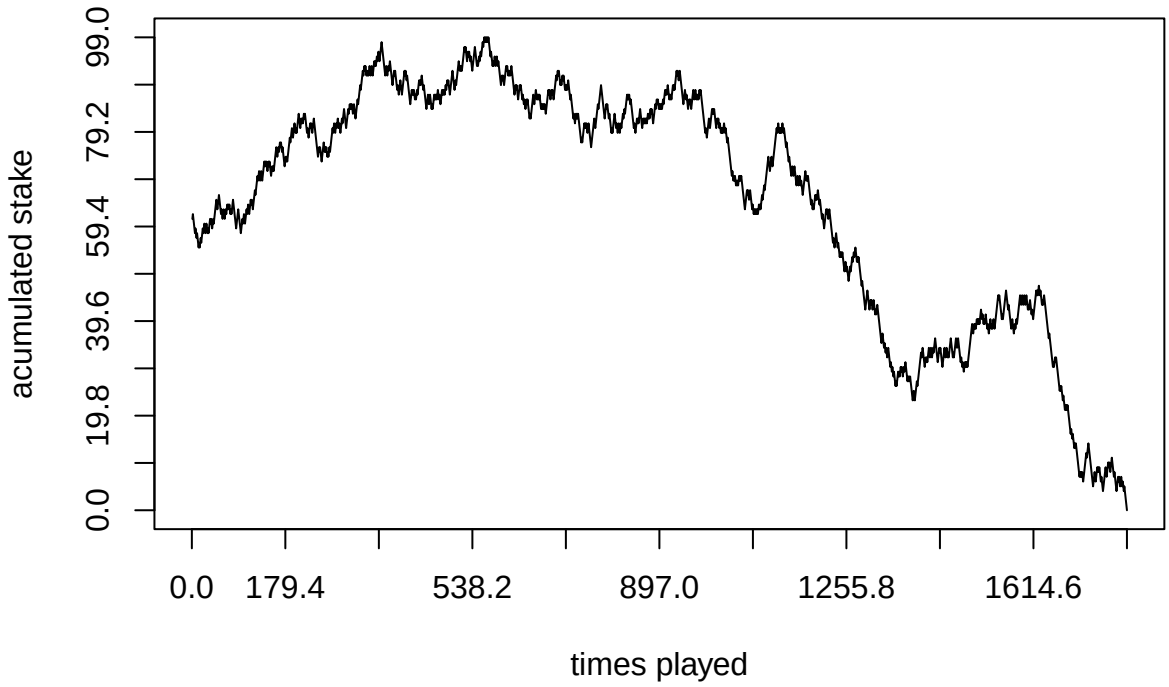
```
# gamblersruin.R
# Example 1.11

# gamble(k, n, p)
  #   k: Gambler's initial state
```

```r
  #   n: Gambler plays until either $n or Ruin
  #   p: Probability of winning $1 at each play
  #   Function returns 1 if gambler is eventually ruined
  #                    returns 0 if gambler eventually wins $n

gamble <- function(k, n, p, goal) {

    stake <- k
    stake_historic <- integer()
    reach_goal <- FALSE
    i <- 1

    while (stake > 0 & stake < n) {
        bet <- sample(c(-1, 1), 1, prob = c(1 - p, p))
        stake <- stake + bet
        stake_historic[i] <- stake
        if (stake == goal)
          reach_goal = TRUE
        i <- i + 1
    }

    times_played <- length(stake_historic)
  max_stake <- max(stake_historic)
  cat("Times played: ", times_played, "\tMaximum acumulated stake: ", max_stake)
  plot(stake_historic, xlab="times played", ylab= "acumulated stake", cex=1, type="l",
       xlim=c(0, times_played), ylim=c(0, max_stake), lty="solid",
       xaxp=c(0, times_played, 10), yaxp=c(0, max_stake, 10))

    if (stake == 0)
       return(list(1, reach_goal))
    else
      return(list(0, reach_goal))
}

k <- 60
n <-  1000 # Let's assign a big number to represent that the rival has so much money
p <- 0.50
goal <- 100
trials <- 10

res <- replicate(trials, gamble(k, n, p, goal))
```

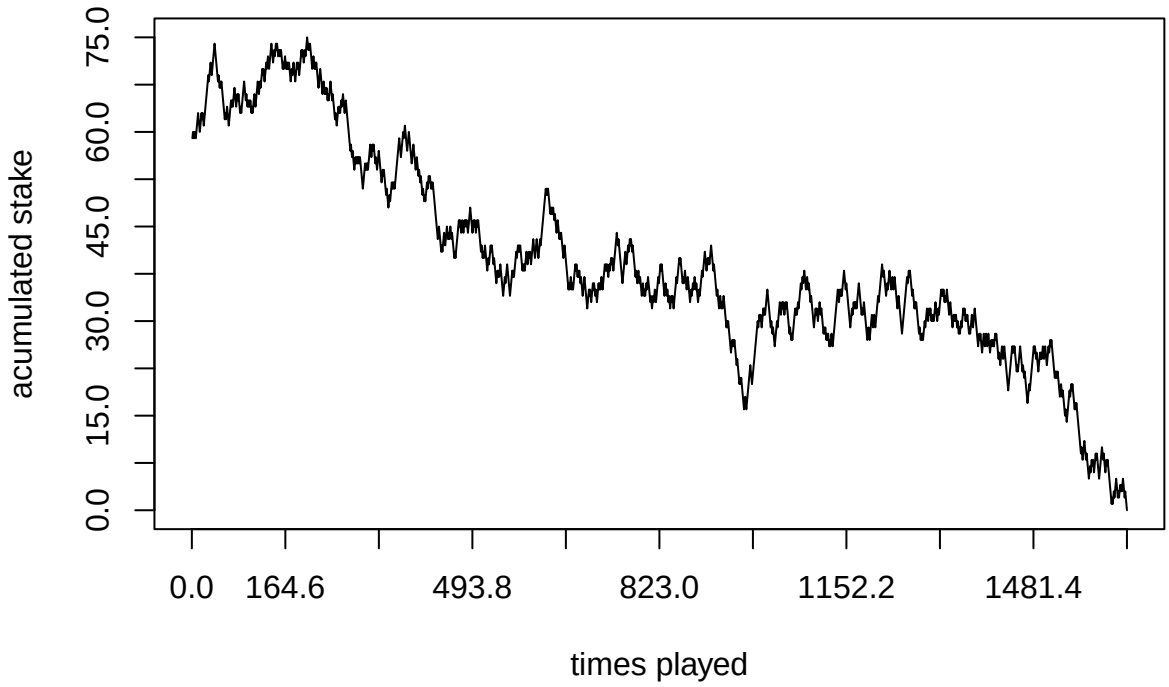
```
## Times played:  4104  Maximum acumulated stake:  135
```



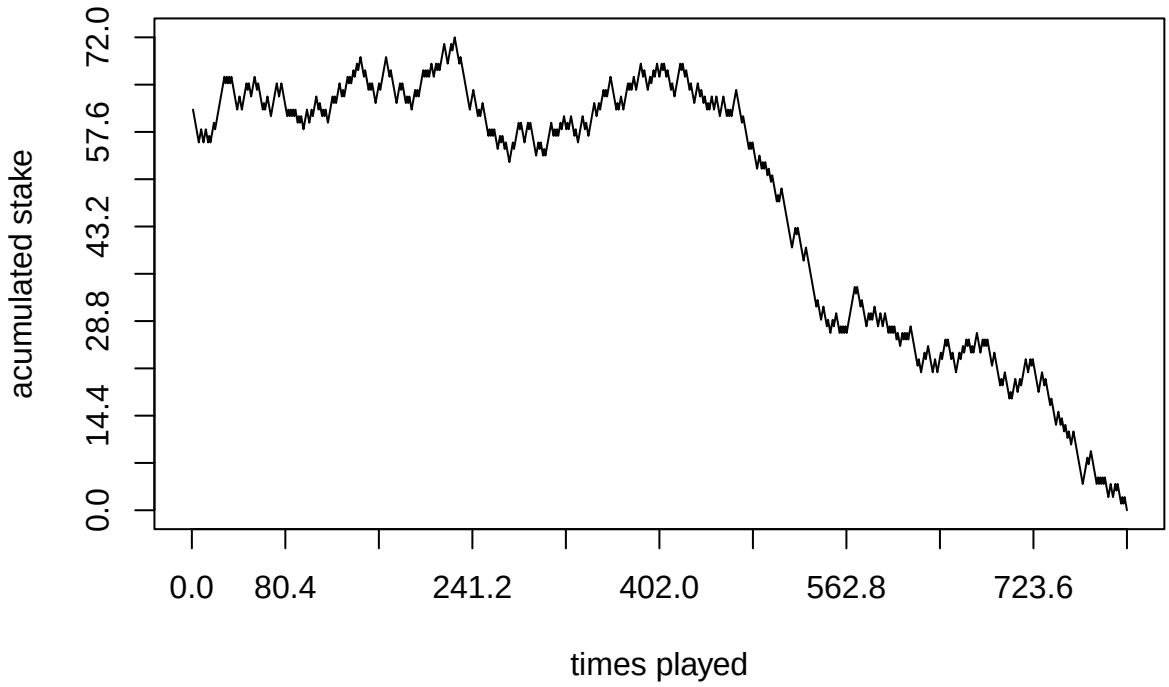

```
## Times played:  1794  Maximum acumulated stake:  99
```



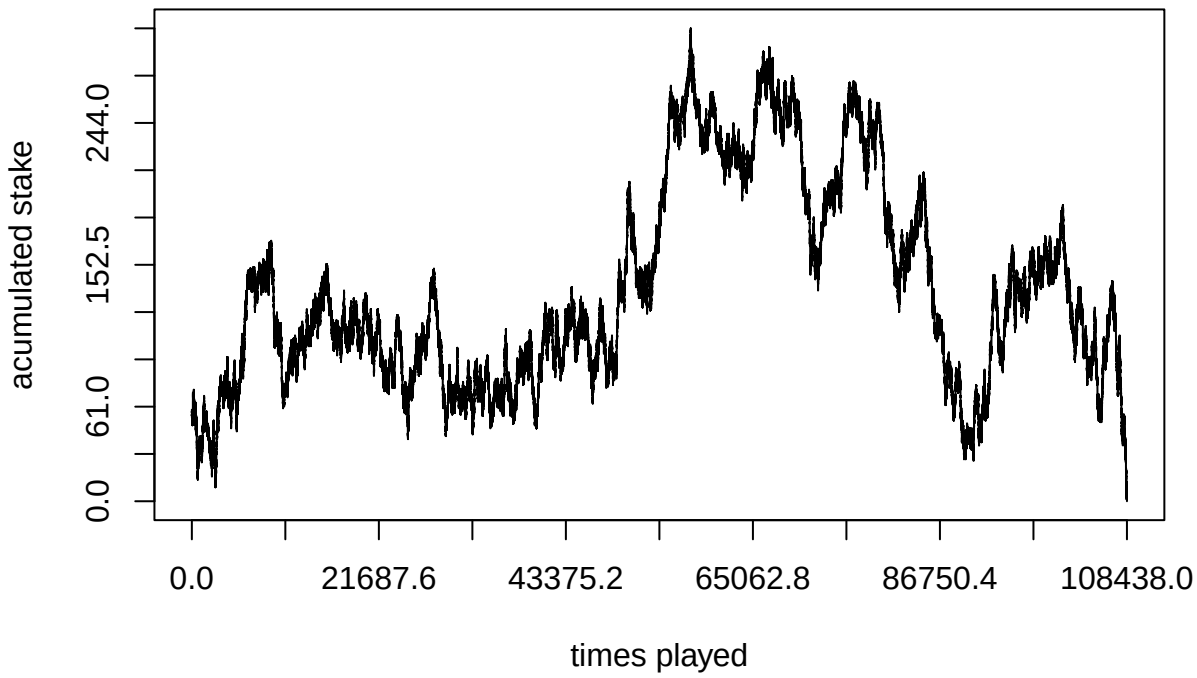

```
## Times played:  1646  Maximum acumulated stake:  75
```



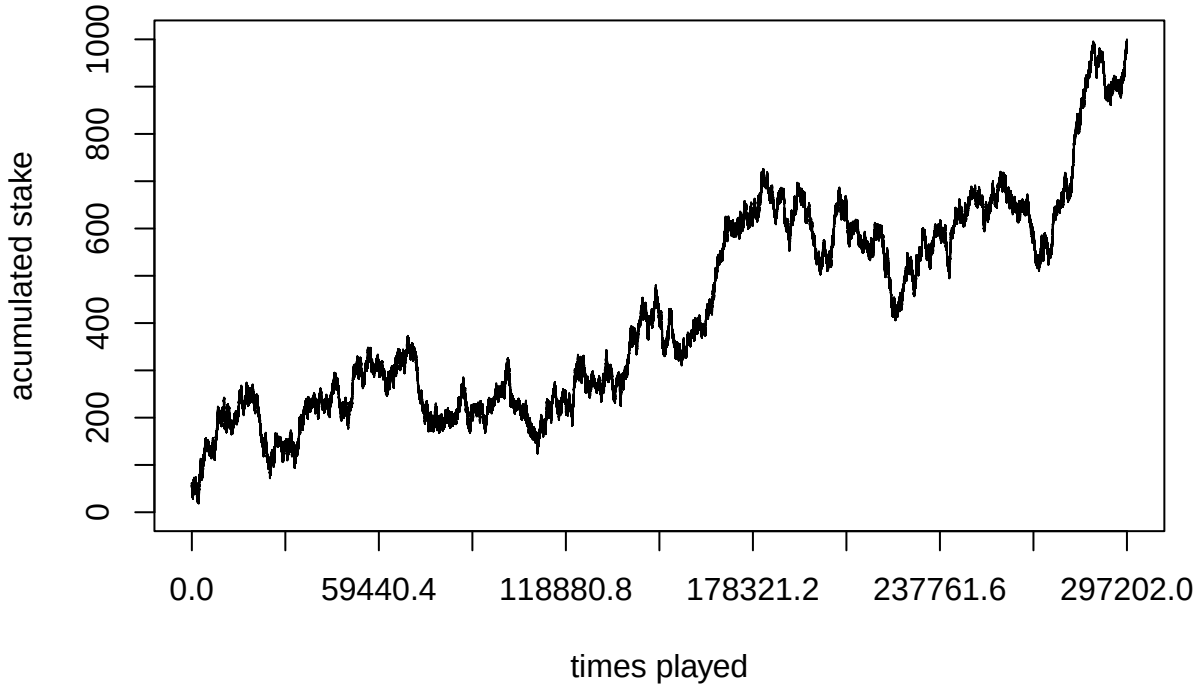

```
## Times played:  804   Maximum acumulated stake:  72
```



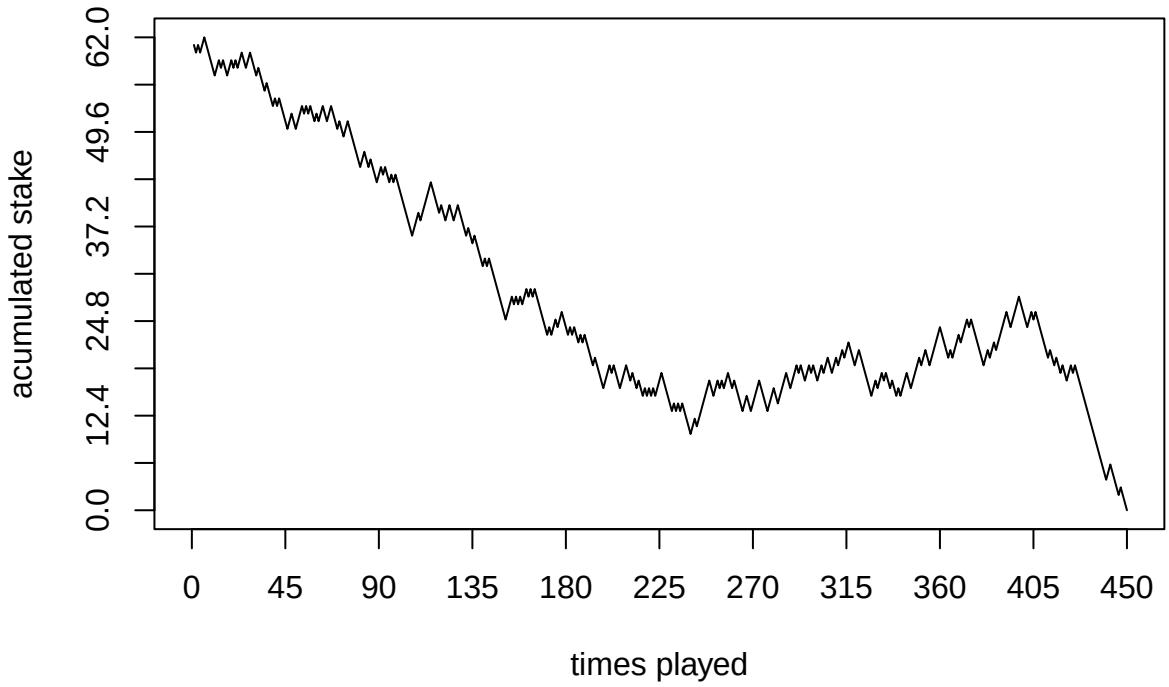

```
## Times played:  108438    Maximum acumulated stake:  305
```



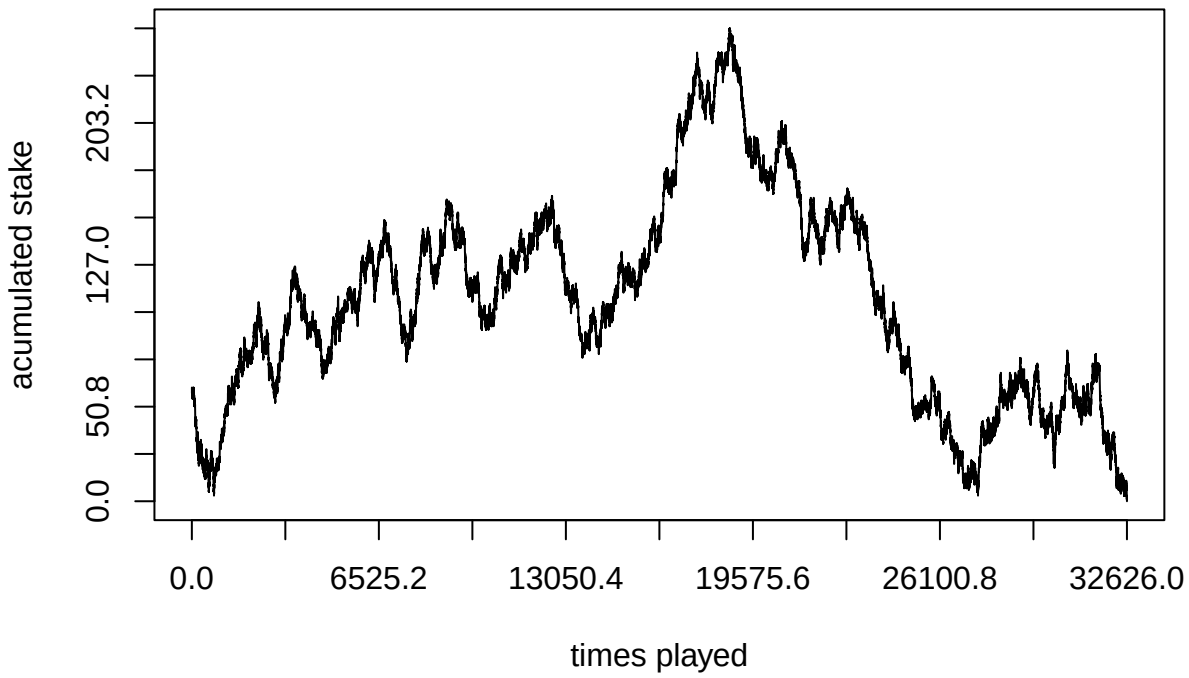

```
## Times played:  297202    Maximum acumulated stake:  1000
```



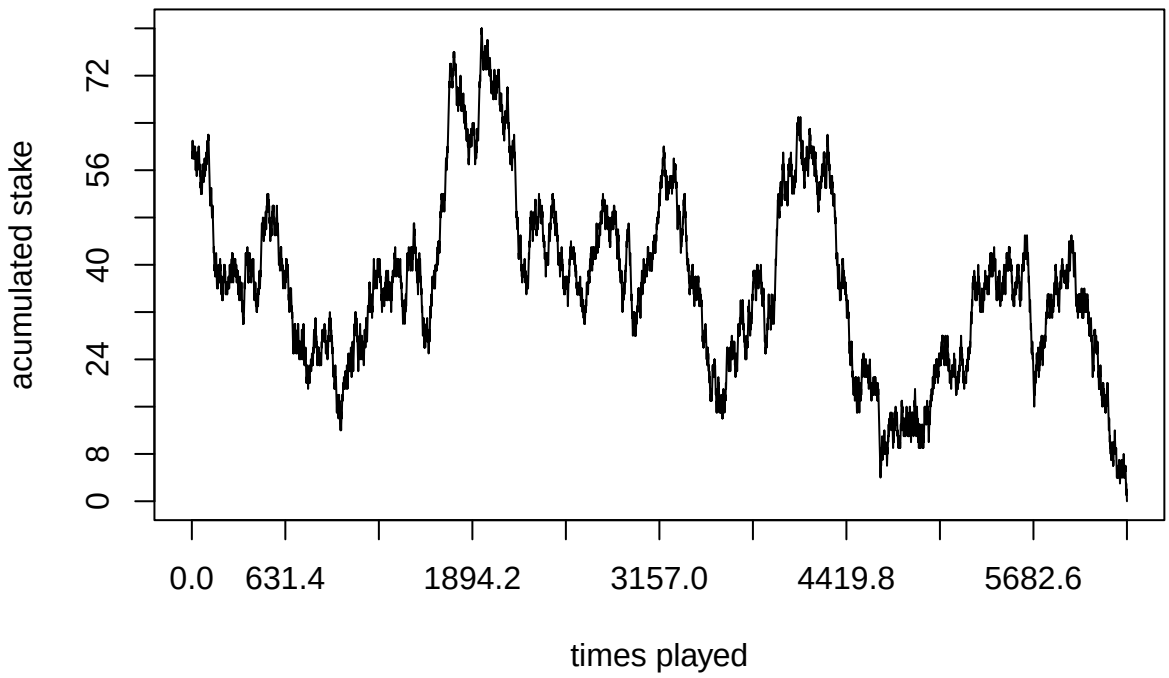

```
## Times played:  450   Maximum acumulated stake:  62
```



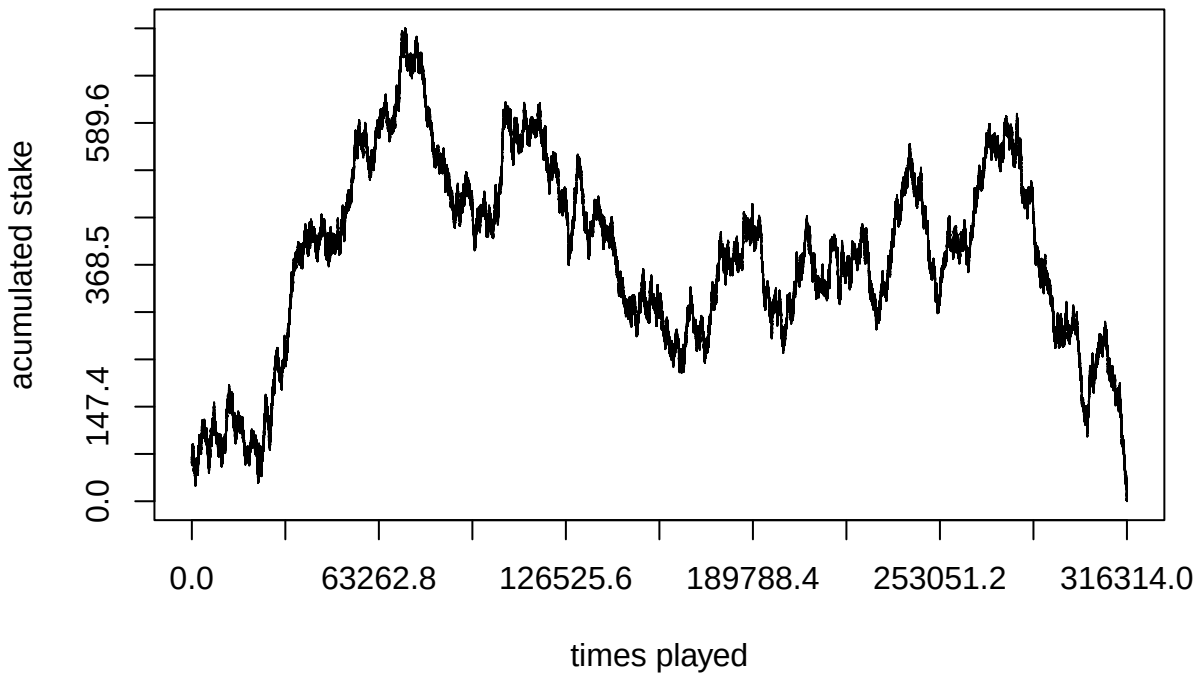

```
## Times played:  32626     Maximum acumulated stake:  254
```




```
## Times played:  6314  Maximum acumulated stake:  80
```




```
## Times played:  316314    Maximum acumulated stake:  737
```



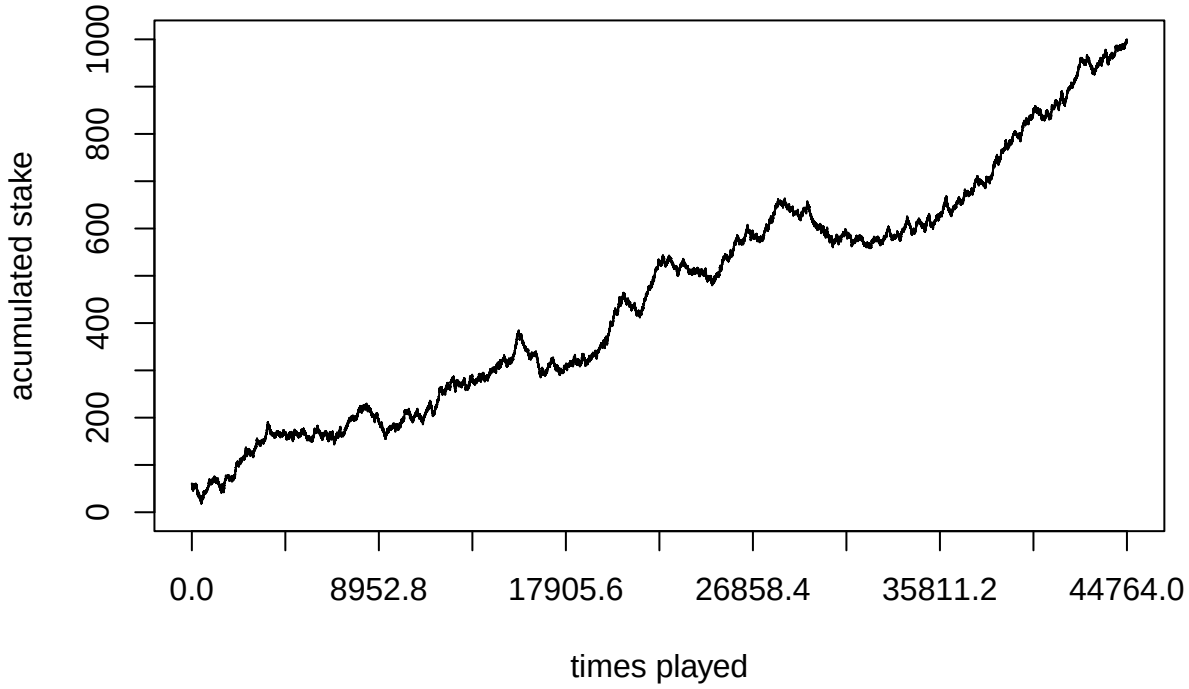

```
res_reach_goal <- res[2, res[1,] == 1 & res[2,] == TRUE]
prob_reach_goal <- length(res_reach_goal) / trials

cat("The probability of reaching the goal of $", goal,
    "before lossing everything is", prob_reach_goal, ", with p =", p)
```

```
## The probability of reaching the goal of $ 100 before lossing everything is 0.4 , with p = 0.5
```

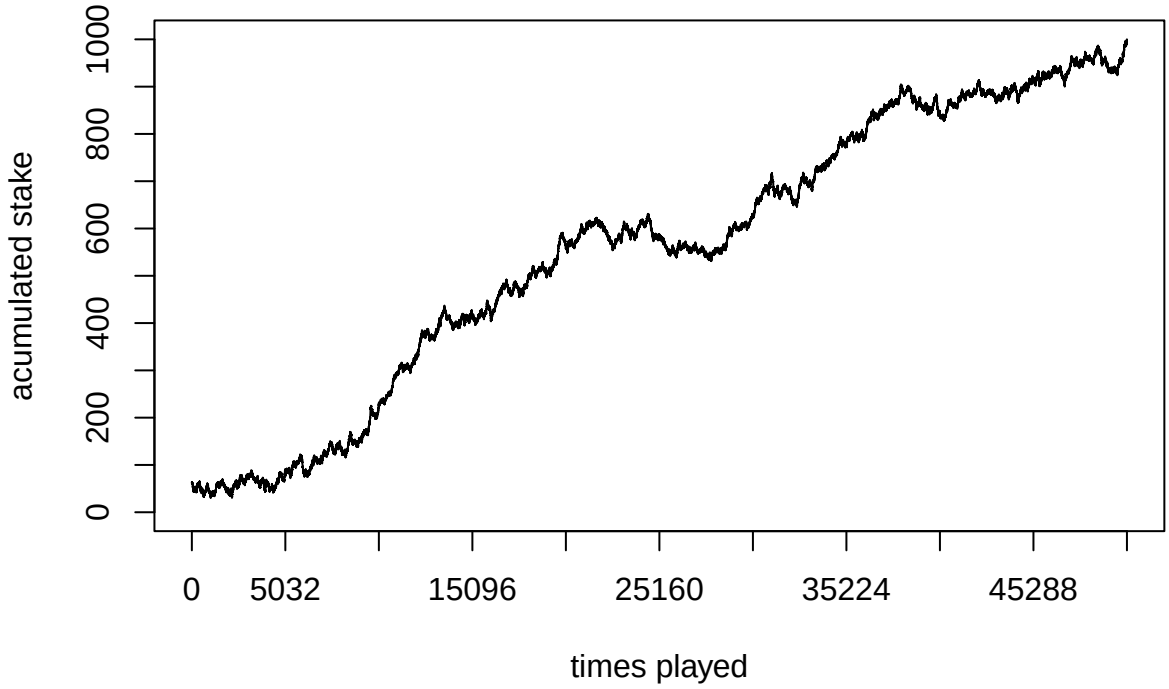
(b) The gambler wins each round with probability p = 0.51. Simulate the probability the gambler wins $100 before he loses everything.

```
k <- 60
n <-  1000 # Let's assign a big number to represent that the rival has so much money
p <- 0.51
goal <- 100
trials <- 10

res <- replicate(trials, gamble(k, n, p, goal))
```

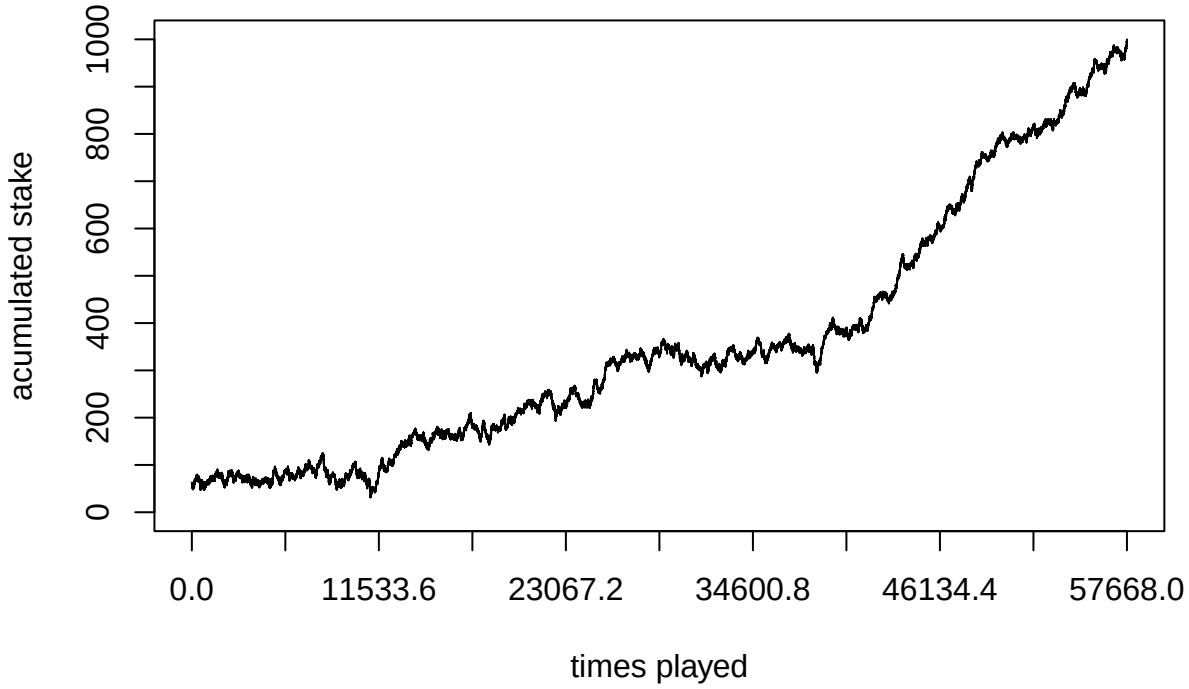
```
## Times played:  44764     Maximum acumulated stake:  1000
```



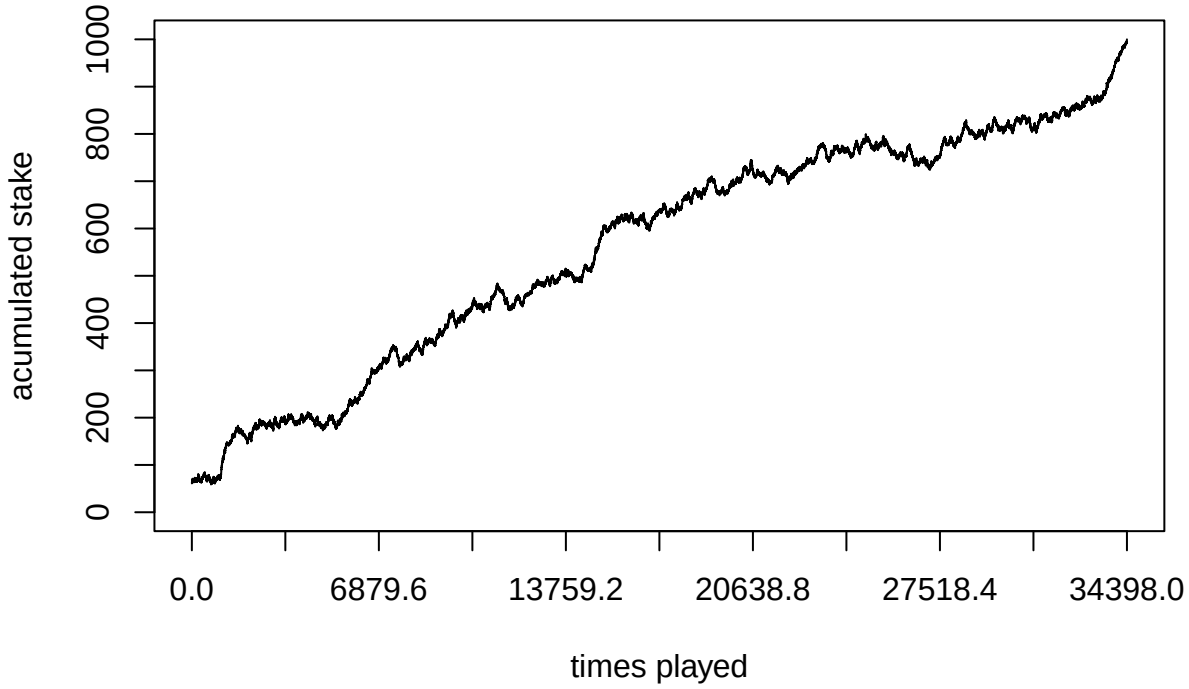

```
## Times played:  50320     Maximum acumulated stake:  1000
```



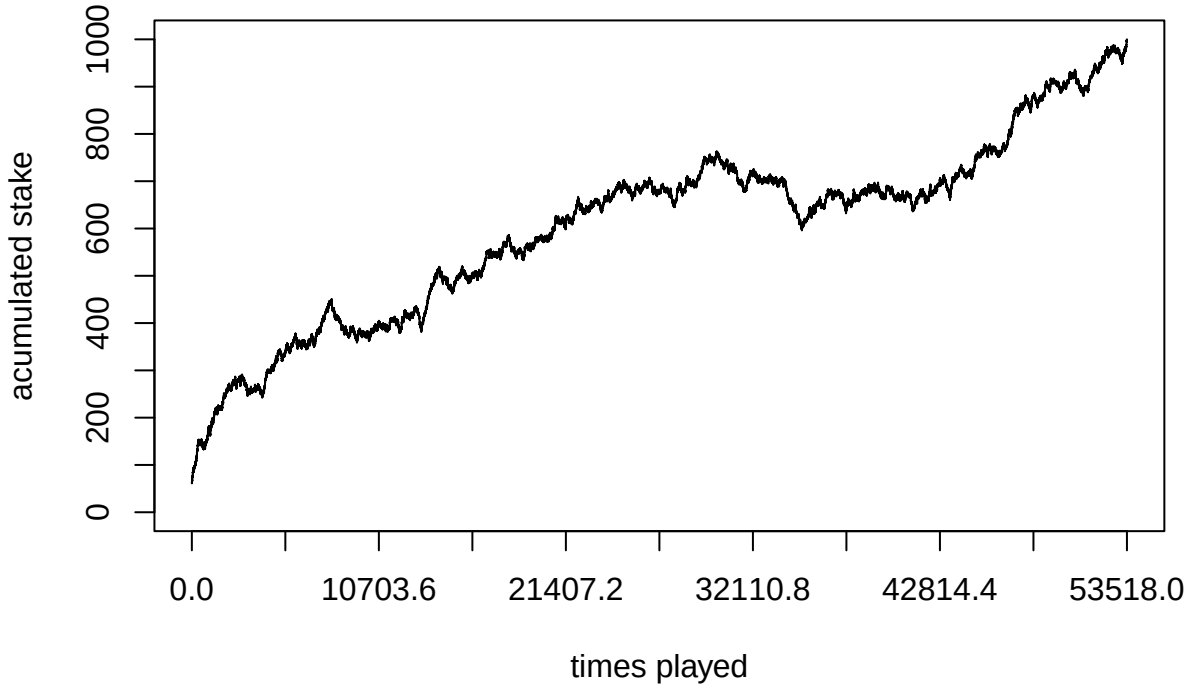

```
## Times played:  57668     Maximum acumulated stake:  1000
```



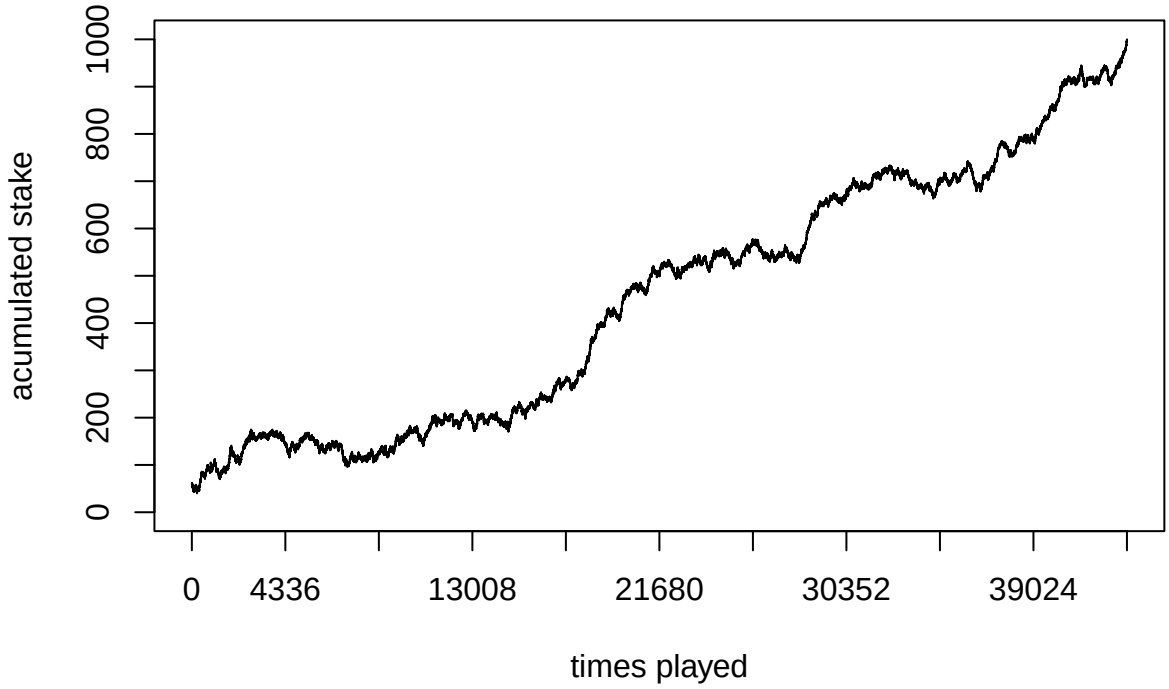

```
## Times played:  34398     Maximum acumulated stake:  1000
```



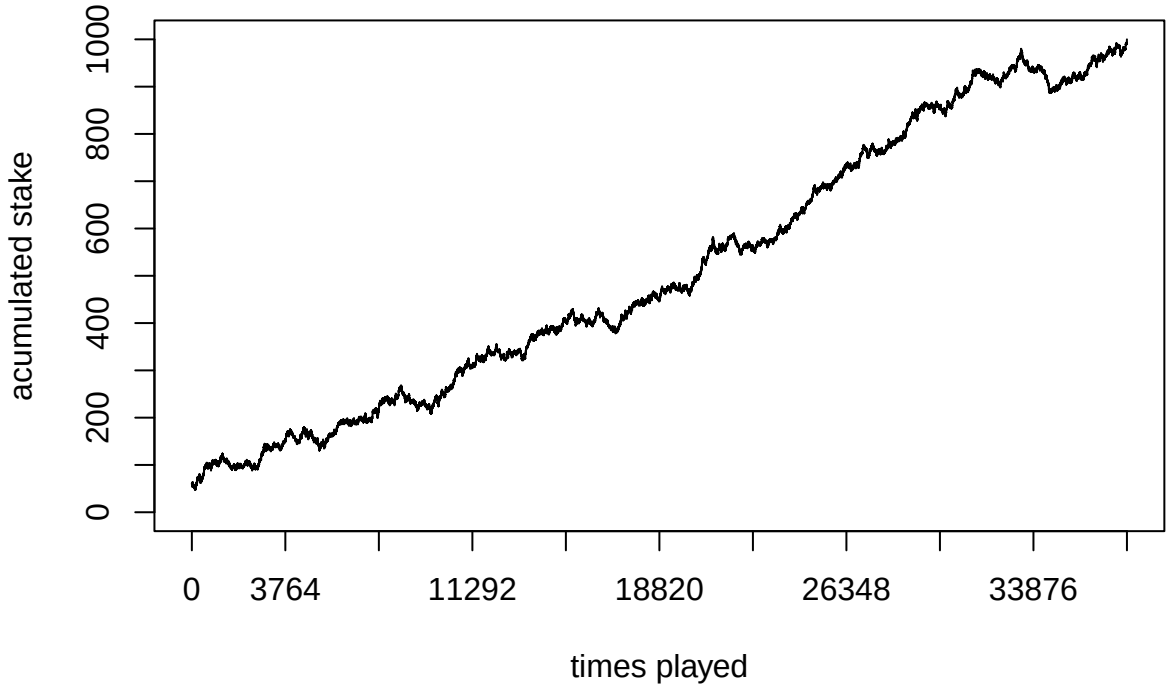

```
## Times played:  53518     Maximum acumulated stake:  1000
```



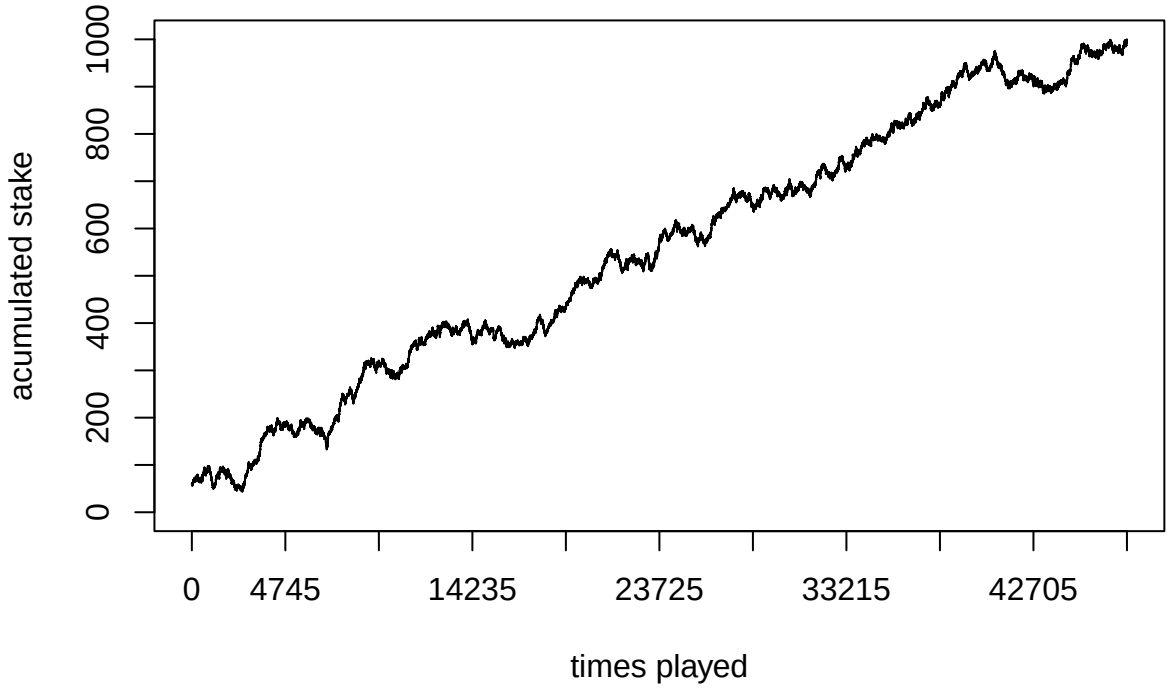

```
## Times played:  43360     Maximum acumulated stake:  1000
```



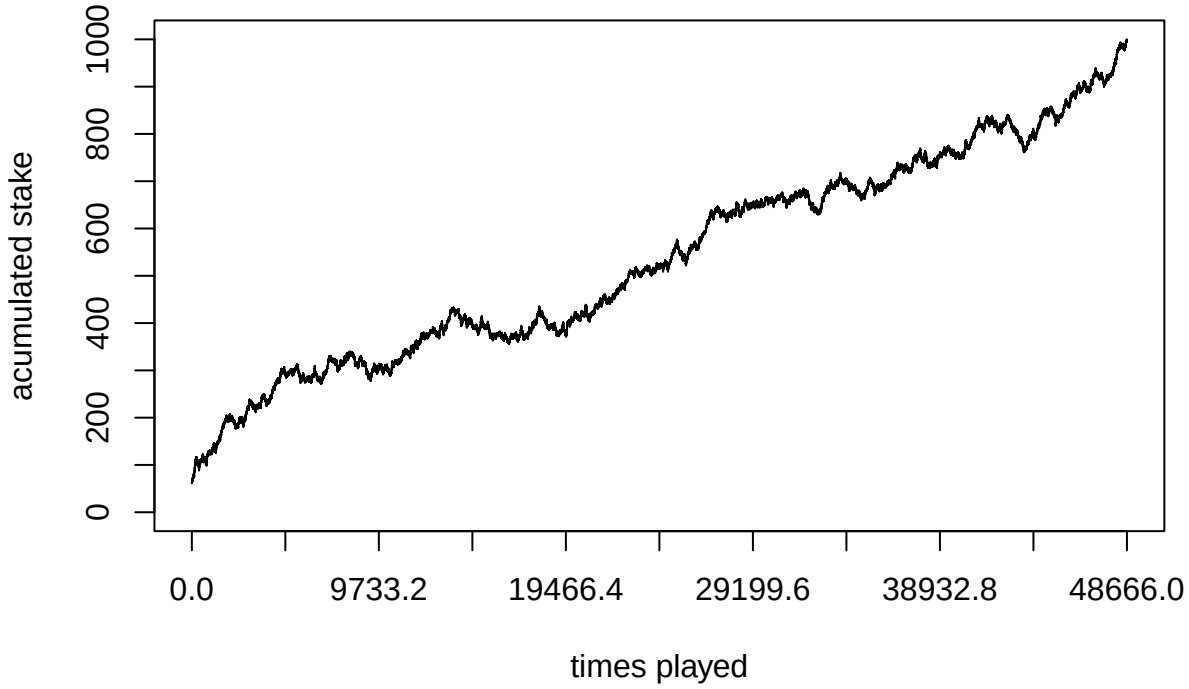

```
## Times played:  37640     Maximum acumulated stake:  1000
```



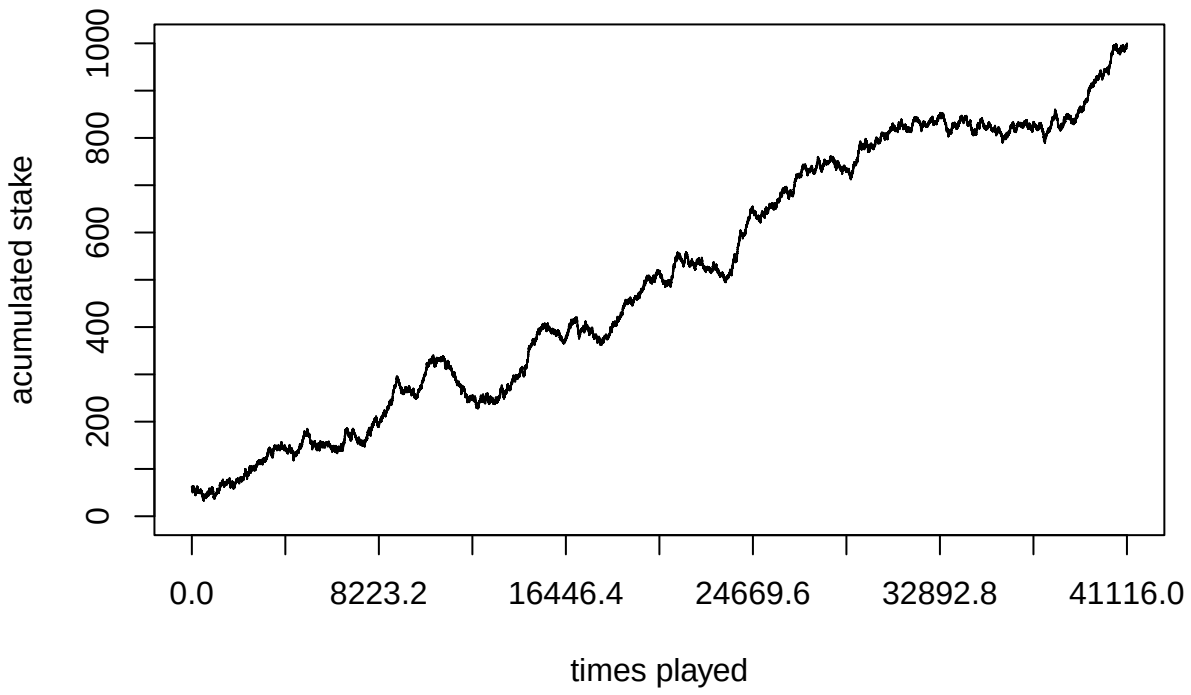

```
## Times played:  47450     Maximum acumulated stake:  1000
```




```
## Times played:  48666     Maximum acumulated stake:  1000
```




```
## Times played:  41116     Maximum acumulated stake:  1000
```




```
res_reach_goal <- res[2, res[1,] == 1 & res[2,] == TRUE]
prob_reach_goal <- length(res_reach_goal) / trials

cat("The probability of reaching the goal of $", goal,
    "before lossing everything is", prob_reach_goal, ", with p =", p)
```

```
## The probability of reaching the goal of $ 100 before lossing everything is 0 , with p = 0.51
```

1.36) See Example 1.2 and the script file ReedFrost.R. Observe the effect on the course of the disease by changing the initial values for the number of people susceptible and infected. How does increasing the number of infected people affect the duration of the disease?

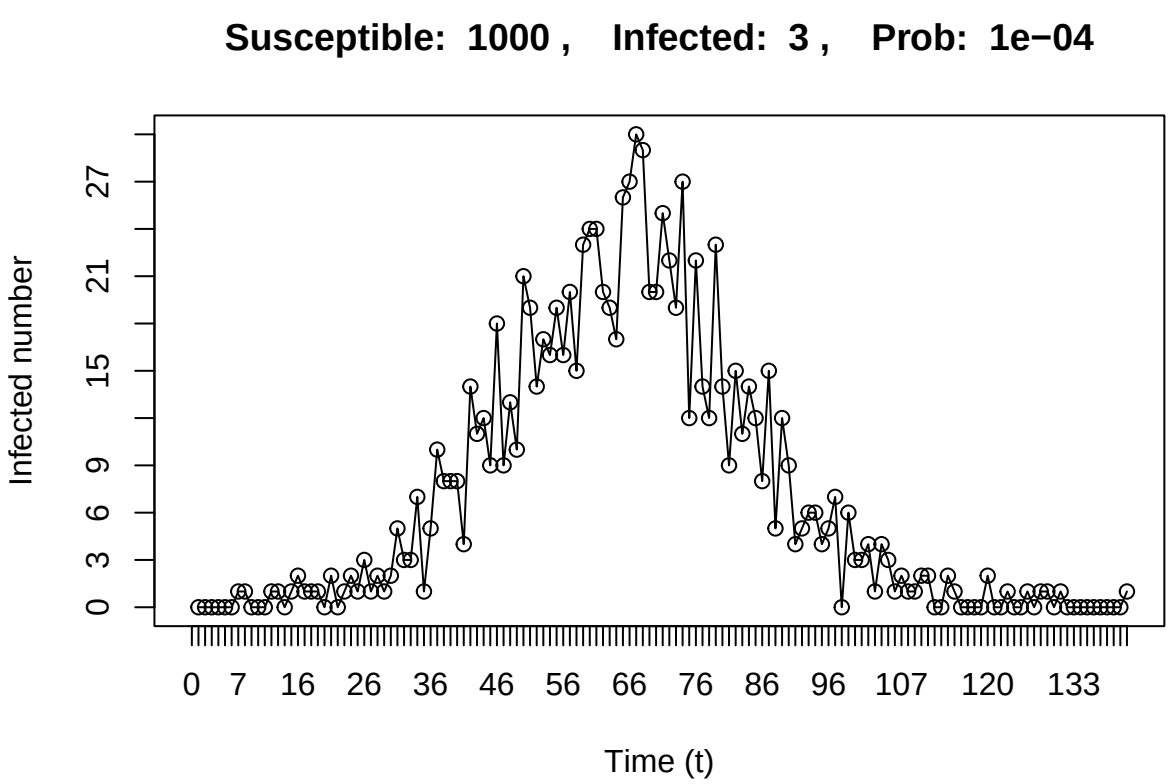
So first, we define our functions that simulate the evolution of the susceptible and infected people.

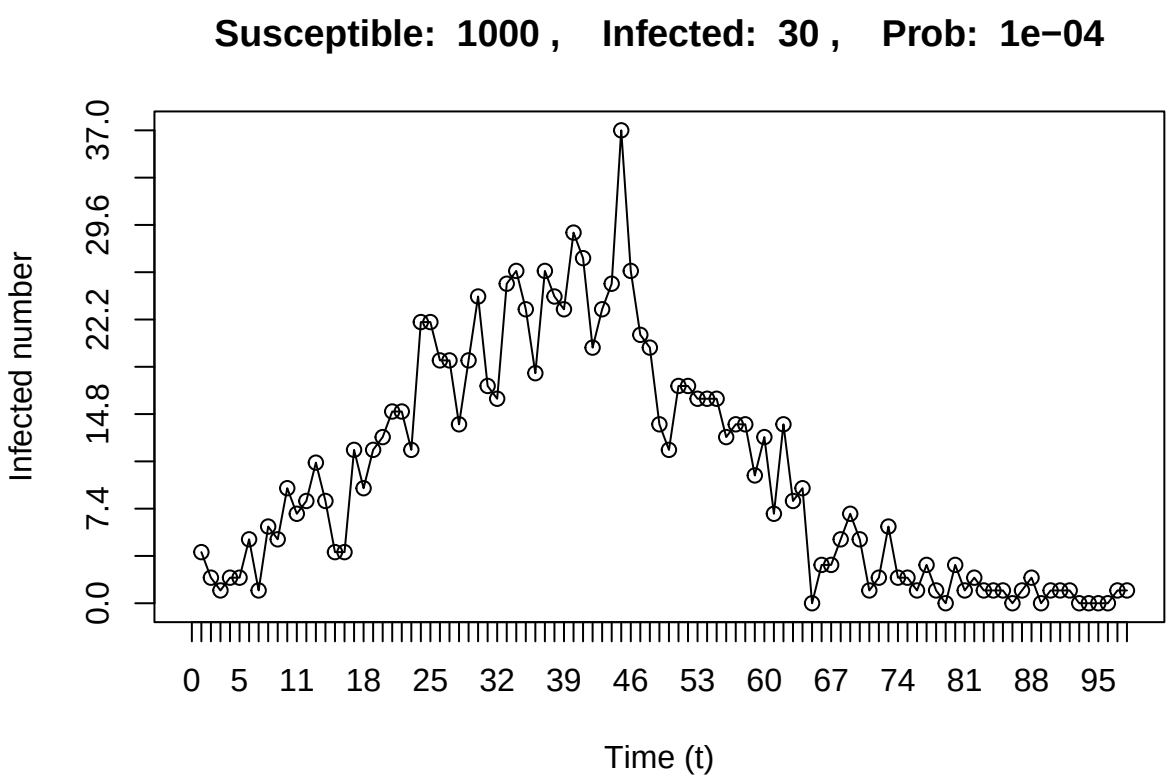
```
 # SIR model
 # SIR(S0, I0, p)
   # S0 : Initial susceptible population
   # I0 : Initial infected population
   # p  : Probability of getting infected

SIR<- function(S0, I0, p){

  S0_aux <- S0
```

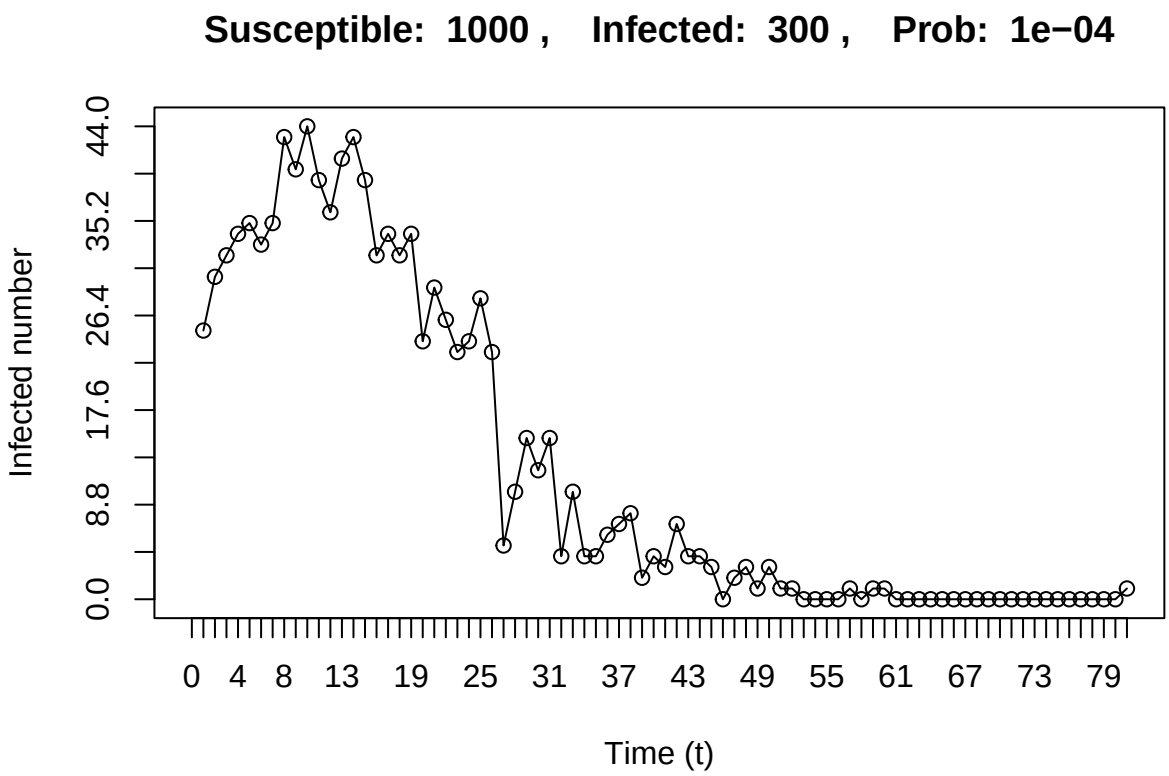
```
  I0_aux <- I0

  r1 <- list()
  X <- list()
  Y <- list()
  W <- list()
  i <- 1

  while (S0 > 0){
    C <- rbinom(S0, I0, p)
    I1 <- length(C[C > 0])
    S0 <- S0 - I1
    I0 <- I0 + I1
    X[i] <- S0
    Y[i] <- I0
    W[i] <- I1
    i <- i + 1
  }

  r1<- list()
  r1[[1]]<- X
  r1[[2]]<- Y
  r1[[3]]<- W

  W_aux <- unlist(W)
  time_limit <- length(W_aux)
  infected_limit <- max(W_aux)
  title <- paste("Susceptible: ", S0_aux, ",    Infected: ", I0_aux, ",    Prob: ", p)
  plot(W_aux, xlab="Time (t)", ylab= "Infected number",
       cex=1, type="o", xlim=c(0, time_limit), ylim=c(0, infected_limit), lty="solid",
       xaxp=c(0, time_limit, time_limit), yaxp=c(0, infected_limit, 10),
       main = title)

  return(r1)
}
```

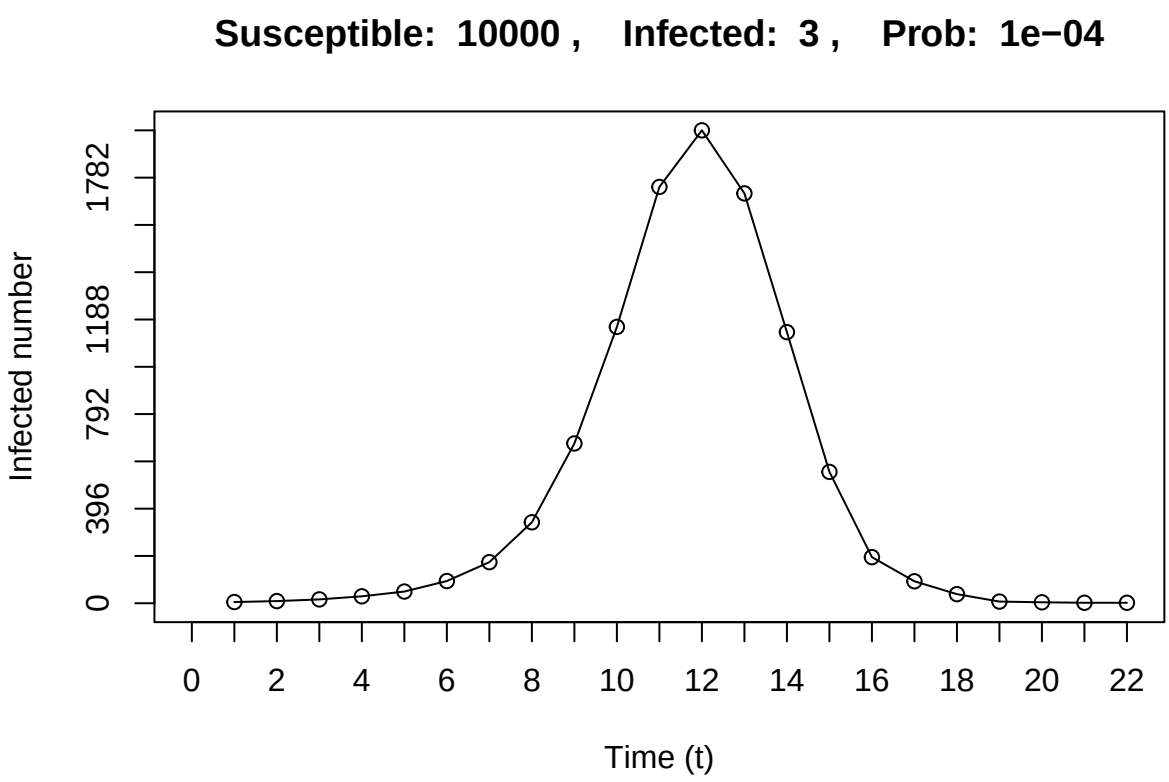
Now, we are going to experiment with various parameters as it's shown below:

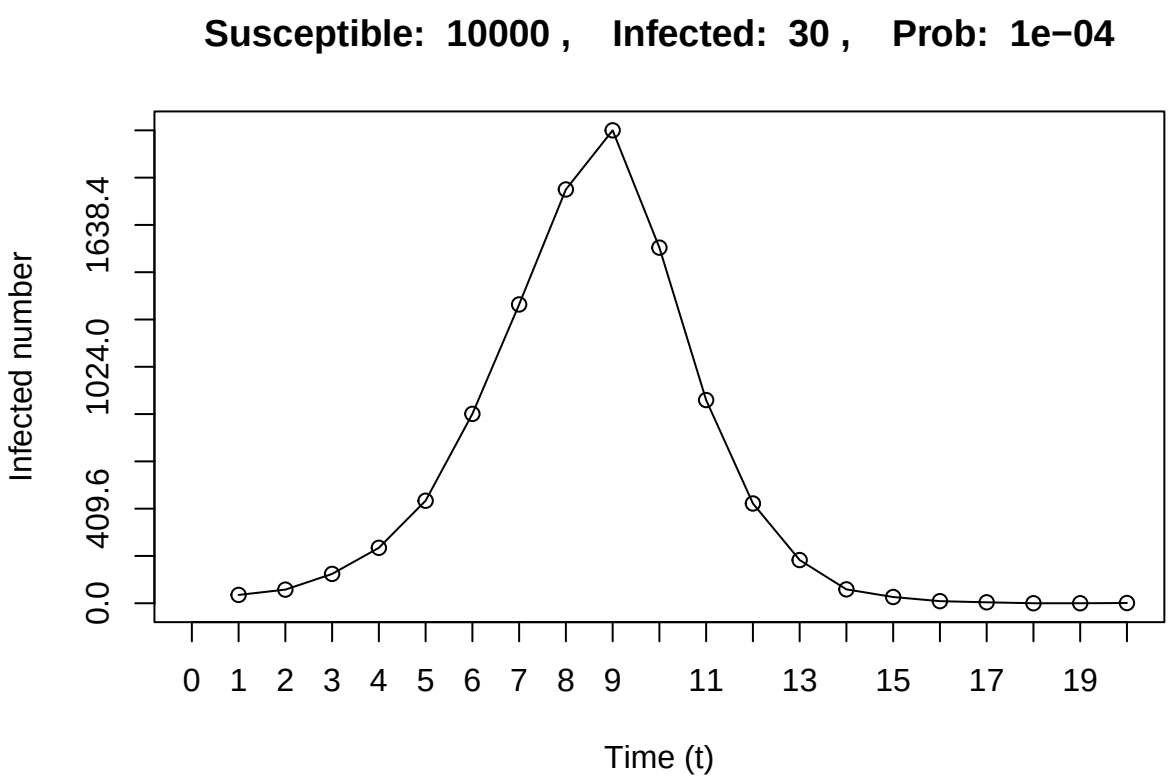
```
S0_vector <- c(1000, 10000, 100000)
I0_vector <- c(3, 30, 300)
p_vector <- c(0.0001)

for (i in 1:length(S0_vector)) {
  for (j in 1:length(I0_vector)) {
    for (k in 1:length(p_vector)) {
      SIR(S0_vector[i], I0_vector[j], p_vector[k])
    }
  }
}
```

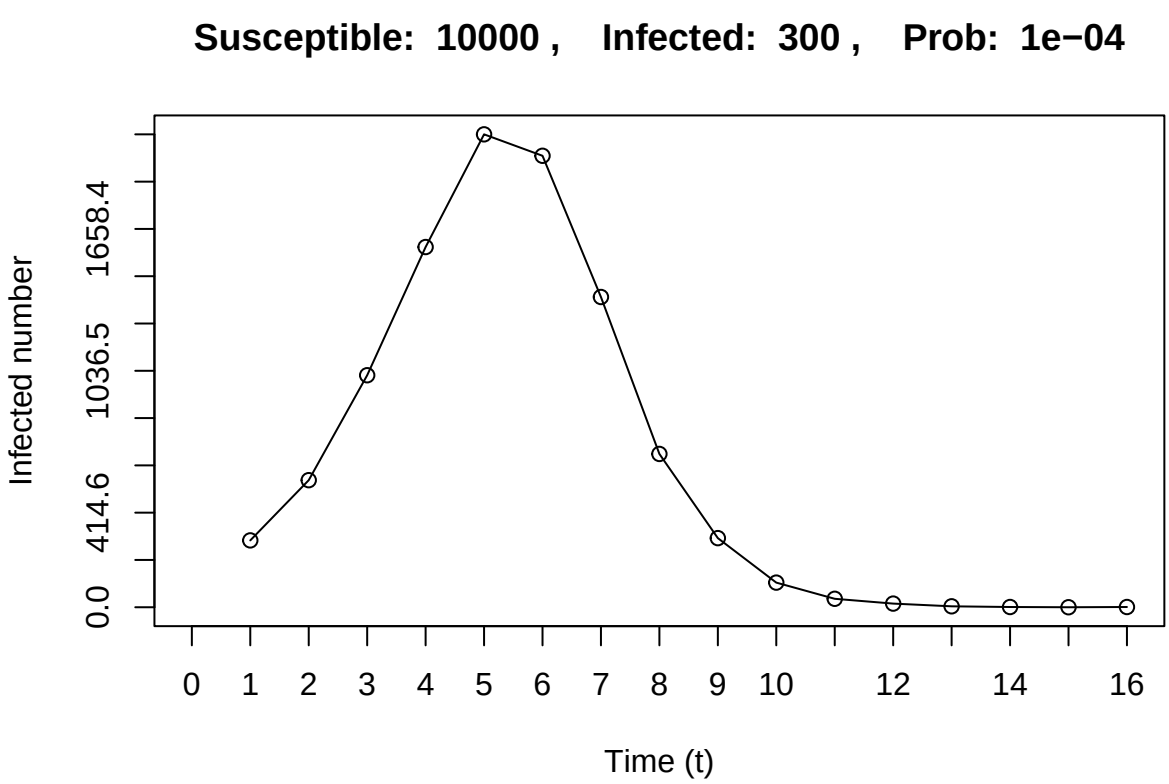

Susceptible: 1000 , Infected: 3 , Prob: 1e−04
Infected number
0
3
6
9
15
21
27
0
7
16
26
36
46
56
66
76
86
96
107
120
133
Time (t)

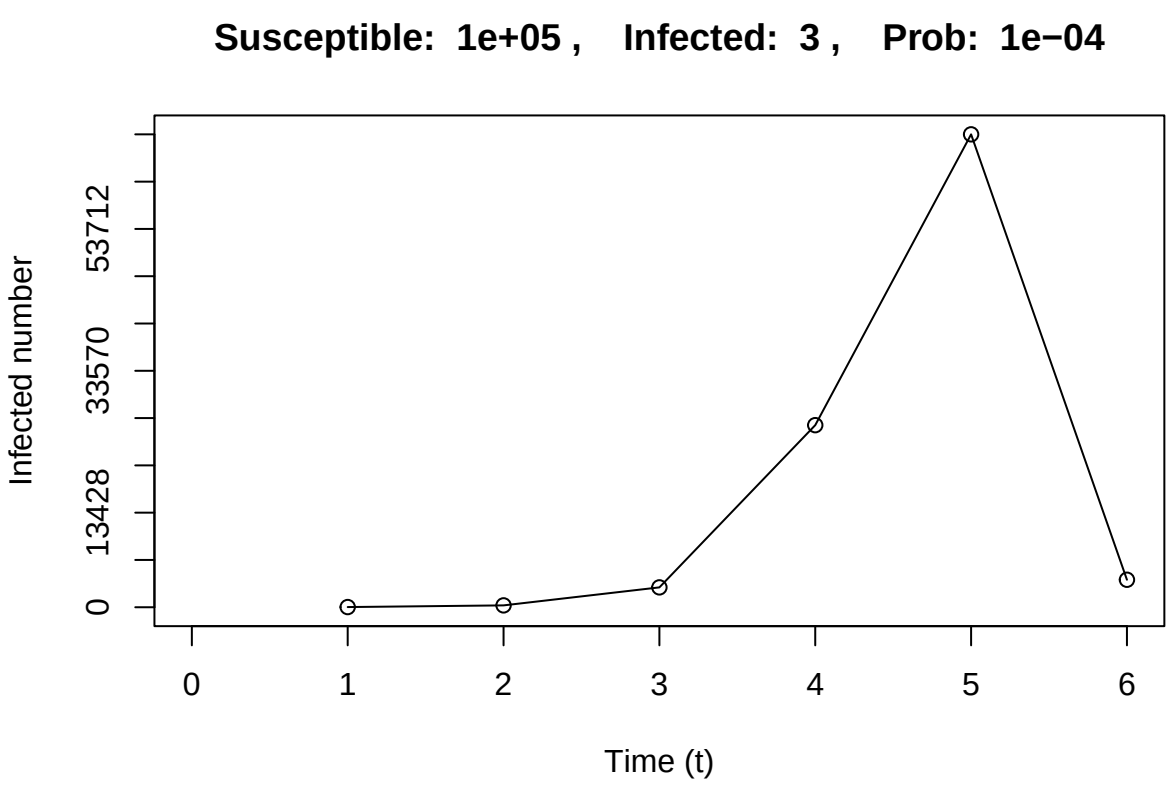

Susceptible: 1000 , Infected: 30 , Prob: 1e−04
Infected number
0.0
7.4
14.8
22.2
29.6
37.0
0 5 11 18 25 32 39 46 53 60 67 74 81 88 95
Time (t)

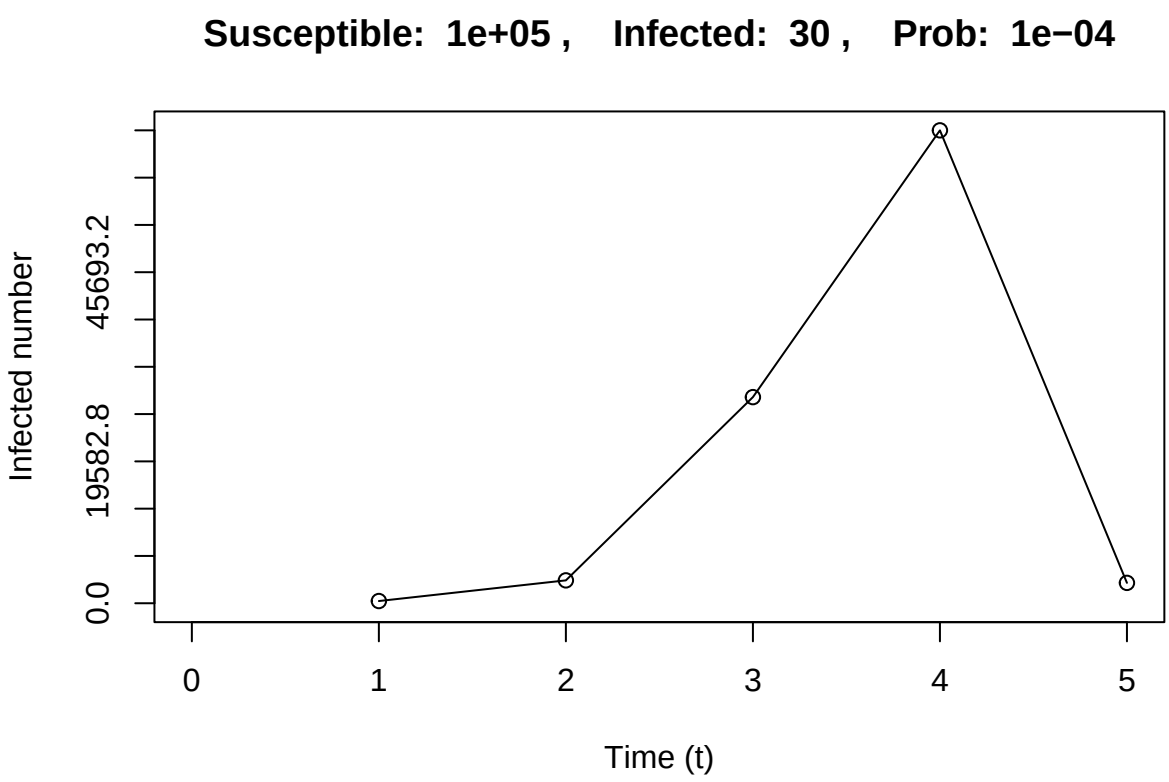

Susceptible: 1000 , Infected: 300 , Prob: 1e−04
Infected number
0.0
8.8
17.6
26.4
35.2
44.0
0
4
8
13
19
25
31
37
43
49
55
61
67
73
79
Time (t)

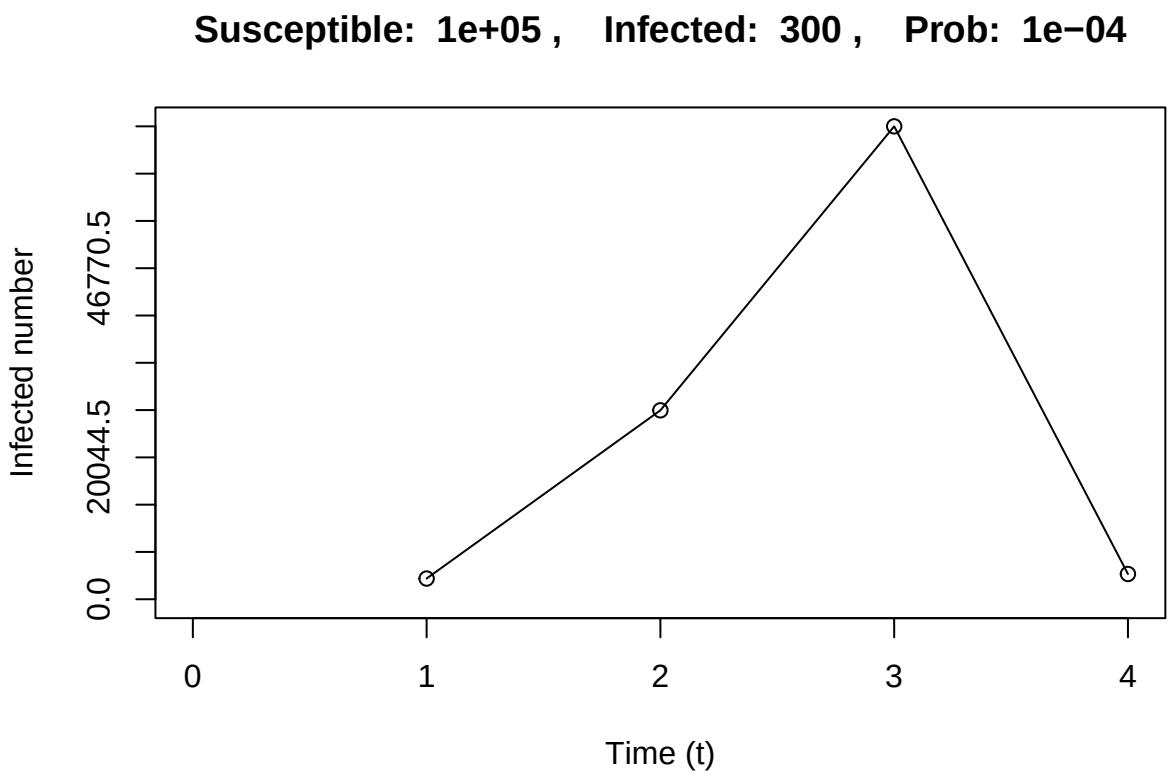

Susceptible: 10000 , Infected: 3 , Prob: 1e−04
Infected number
0
396
792
1188
1782
0
2
4
6
8
10
12
14
16
18
20
22
Time (t)


Susceptible: 10000 , Infected: 30 , Prob: 1e−04
Infected number
0.0
409.6
1024.0
1638.4
0 1 2 3 4 5 6 7 8 9 11 13 15 17 19
Time (t)


Susceptible: 10000 , Infected: 300 , Prob: 1e−04
Infected number
0.0
414.6
1036.5
1658.4
0
1
2
3
4
5
6
7
8
9
10
12
14
16
Time (t)


Susceptible: 1e+05 , Infected: 3 , Prob: 1e−04
Infected number
0
13428
33570
53712
0
1
2
3
4
5
6
Time (t)


Susceptible: 1e+05 , Infected: 30 , Prob: 1e−04
Infected number
0.0
19582.8
45693.2
0
1
2
3
4
5
Time (t)



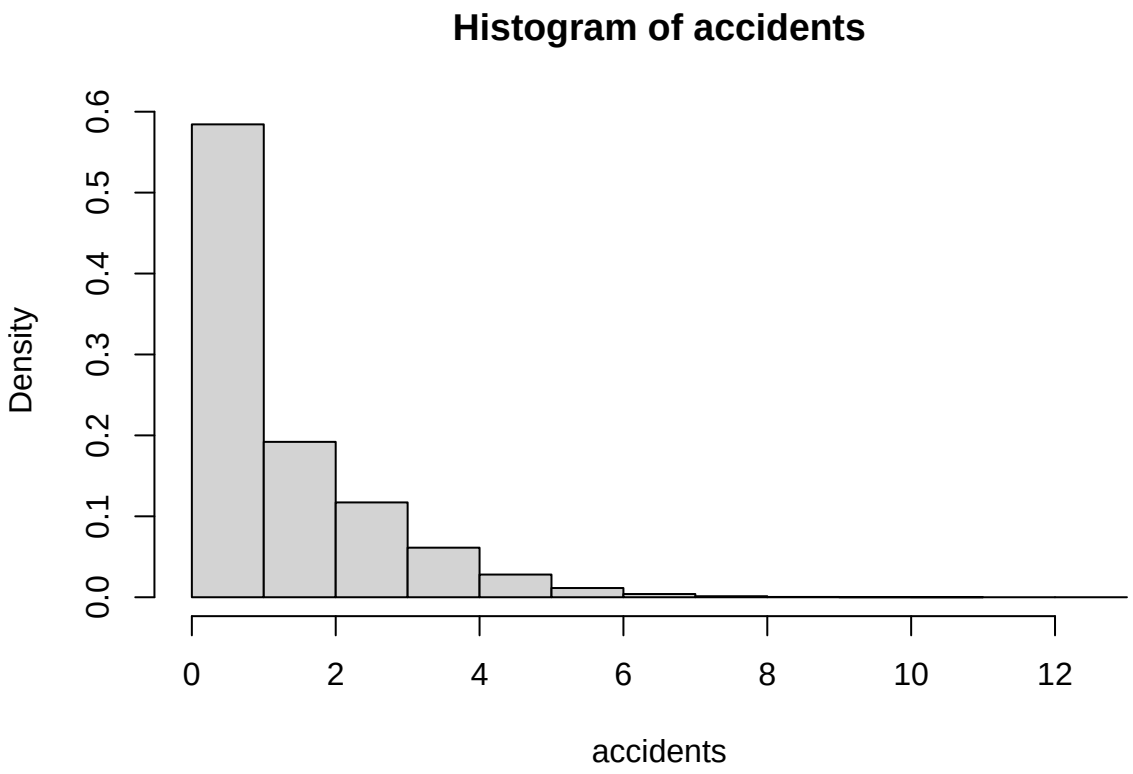
With the previous graphics we can remark that the more the number of infected people increases, the more it reduces the duration of the disease. Similarly, it's worth mentioning that if we increase the probability of getting infected, people get infected faster therefore the global infection on the population ends considerably faster.

1.37) Simulate the results of Exercise 1.28. Estimate the mean and variance of the number of accidents per day.

So, exercise 1.28 says the following: *On any day, the number of accidents on the highway has a Poisson distribution with parameter Λ. The parameter Λ varies from day to day and is itself a random variable. Find the mean and variance of the number of accidents per day when Λ is uniformly distributed on (0, 3).*

Therefore, we are going to use simulate an uniform distribution with parameters a=0 and b=3 within the function runif given by R. And then use that result on the function rpois in order to make random generation of the poisson distribution.

```
n <- 1000000
a <- 0
b <- 3

set.seed(1)
rand_lambda <- runif(n, min = a, max = b)
accidents <- rpois(n, rand_lambda)

hist(accidents, breaks = 18, freq = FALSE)
```




```
accidents_mean <- mean(accidents)
accidents_var <- var(accidents)

cat("Mean=", accidents_mean, "\tVariance=", accidents_var)
```

```
## Mean= 1.49949    Variance= 2.254094
```