Final report(Modeling and Simulation)

Surajnath Sidh(U101114FCS146)

17 Apr. 2017

Contents

Snakes and Ladders Game Simulation

Problem Statment

Consider the following snakes-and-ladders game. Let N be the number of tosses to reach the finish using a fair dice. Write a R program to calculate the expectation of N . Also, draw a plot in which the x -axis shows the number of rolls, and the y -axis shows the percentage of games that were completed in that number of rolls.

Code

The ladders and slides/snakes will be set up as a data frame,

```
ladder.df <- data.frame(start=c(3,11), end=c(13,17))
slide.df <- data.frame(start=c(10,16,18), end=c(5,2,8))
```

```
library(knitr)
```

The ladders:

```
kable(ladder.df, align="c")
```

start	end
3	13
11	17

The Snakes:

```r
kable(slide.df, align="c")
```

start	end
10	5
16	2
18	8

```r
library(foreach)
library(doParallel)
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```r
registerDoParallel(cores=4)
```

```r
getDoParWorkers()
```

```
## [1] 4
```

```r
num.iter <- 100 # Number of play throughs aka games

# Get timing as well
stime <- system.time({
  out.seq <- foreach(icount(num.iter), .combine=rbind) %do% {
    curLoc <- 0
    nroll <- 0
    slides <- 0
    ladders <- 0
    # Keep rolling dice and moving until reach 17 or greater ending the game
    while(curLoc < 17) {
      roll <- sample(6,1) # generate random number between [1 to 6]
      curLoc <- curLoc + roll # increase position
      nroll <- nroll + 1 # increase number of rolls
      # Need to check if we landed on a ladder or slide and move forward or back
      if (any(ladder.df$start %in% curLoc)) {
        curLoc <- ladder.df$end[ladder.df$start %in% curLoc]
        ladders <- ladders + 1
      }
      if (any(slide.df$start %in% curLoc)) {
        curLoc <- slide.df$end[slide.df$start %in% curLoc]
        slides <- slides + 1
      }
    }
    # Create output to store, num rolls, num ladders hit, num slides hit
    out.info <- c(nroll, ladders, slides)
  }})[3]
```

Time taken by simulation

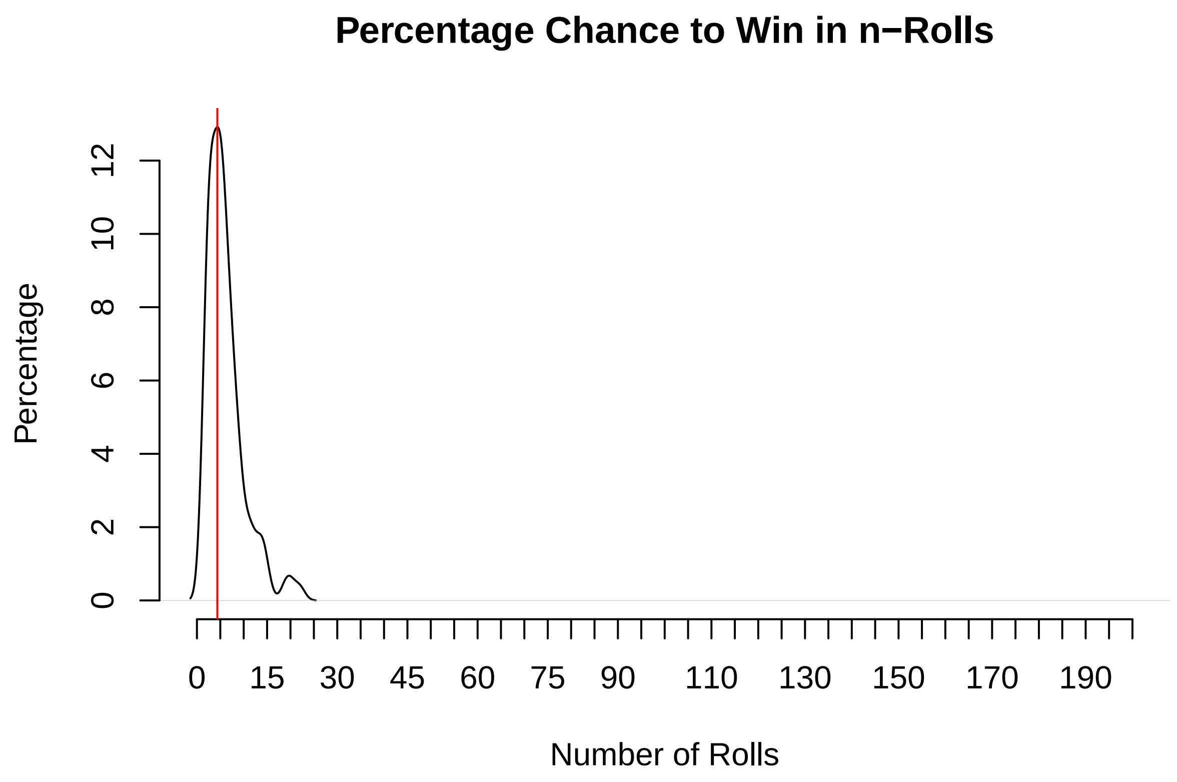
```
stime
```

```
## elapsed
##    0.09
```

Output

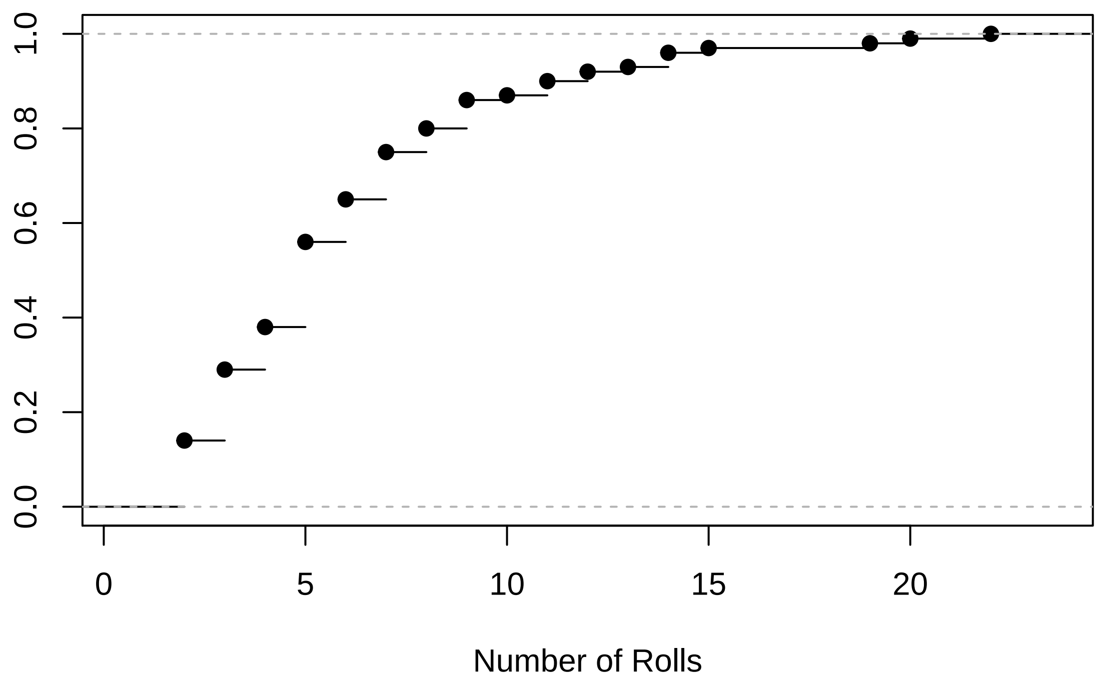
Plot for percentage chance to win the game in n rolls:

```
d <- density(out.seq[,1])
d$y <- d$y * 100
plot(d, main="Percentage Chance to Win in n-Rolls", xlab="Number of Rolls", ylab="Percentage", xlim=c(0
axis(2)
axis(1, at=seq(0,200,5))
abline(v=d$x[which.max(d$y)], col="red")
```



cumulative distribution function:

```
plot(ecdf(out.seq[,1]), xlab="Number of Rolls", ylab="",main="")
```



Conclusion

In this assignment we learned about simulation, system variables, simulation clock etc.

I also created a full scale sankes and ladders simulation which can be found on my github account (https://github.com/electron0zero/R-projects/tree/master/snakes_and_ladders)

Single Server Queuing Model

Problem Statment

Write a R program to simulate a single-server queue, with the following assumptions.

- Assume inter-arrival times are independent and identically distributed (IID) random variables.
- Assume service times are IID, and are independent of inter-arrival times.
- Queue discipline is FIFO.
- Start empty and idle at time 0.
- First customer arrives after an inter-arrival time, not at time 0.
- Stopping rule: When nth customer has completed delay in queue, stop simulation.
- Update system, state variables, clock, event list, statistical counters, all after execution of each event. You are expected to create a function with three arguments

1. Inter arrival rate
2. Service rate, and
3. Stopping rule

Quantities to be estimated are

1. Expected average delay in queue (excluding service time) of the 20 customers completing their delays,
2. Expected average number of customers in queue (excluding any in service), and
3. Expected utilization (proportion of time busy) of the server.

Note: (a) The system begins at time 0 with no customers in the system and an idle server. (b) The times between arrivals are mutually independent and identically distributed exponential random variates with arrival rate lamda = 1 (you can assume initially). (c) The service times are mutually independent and identically distributed exponential random variates with service rate mu = 0.5 (you can assume initially). (d) The server does not take any breaks. (e) A customer departs the system once the service is complete. (f) When 20th customer has completed delay in queue, stop simulation.

Code

Function for SSQ

```
ssq <- function(ra, rs, numOfCustomers) {
  wait_time <- 0;
  total_arrival_time <- 0;
  total_wait_time<- 0;
  total_idle_time <- 0;
  prvious_service_time <- 0;
  for (i in 1:numOfCustomers)
  {
    serTime <- rexp(1, rs);
    interArrTime <- rexp(1, ra);
    total_arrival_time <- total_arrival_time + interArrTime;
    wait_time <- wait_time - interArrTime + prvious_service_time;
    prvious_service_time <- serTime;
    if (wait_time >= 0)
    {
      total_wait_time <- total_wait_time + wait_time
    }
    else
    {
      total_idle_time <- total_idle_time - wait_time;
      wait_time <- 0;
    }
  }
  print(paste("Expected Utilization is",1 - total_idle_time/total_arrival_time,sep = " "))
  print(paste("Expected Average delay in the queue is",total_wait_time/numOfCustomers,"minutes.", sep =
  print(paste("Expected Average Customers in the queue is",total_wait_time/total_arrival_time, sep = "
}
```

Output

```
# ssq(lamda, mu, number_of_cust)
ssq(1,0.5,20)
```

```
## [1] "Expected Utilization is 0.877585274239261"
## [1] "Expected Average delay in the queue is 12.0254537811245 minutes."
## [1] "Expected Average Customers in the queue is 13.2072541546986"
```

Conclusion

In this assignment we learned about simulation in depath and more about simulation system variables

I also created other implimentation using simmer of this which can be found on my my github account (https://github.com/electron0zero/R-projects/tree/master/single_server_queue)

Simulation of a harbour port

problem Statment

A harbour port has three berths 1, 2 and 3. At any given time Berth1 can accommodate two small ships, or one medium ship. Berth2 and Berth3 can each handle one large ship, two medium ships or four small ships.

The interarrival time of ships is 26 hours, exponentially distributed, and small, medium, and large ships are in the proportions 5:3:2 respectively. Queuing for berths is on a first come first served basis, except that no medium or small ship may go to a berth for which a large ship is waiting, and medium ships have a higher priority than small ships.

Unloading times for ships are exponentially distributed with mean times as follows: small ships, 15 hours; medium ships, 30 hours; and large ships, 45 hours.

The loading times are as follows: - Small ships: 24 ± 6 hours uniformly distributed. - Medium ships: 36 ± 10 hours uniformly distributed. - Large ships: 56 ± 12 hours uniformly distributed.

The tide must be high for large ships to enter or leave Berths 2 and 3. Low tide lasts 3 hours, high tide, 10 hours.

Write a R program to simulate the harbour port and 1. run the simulation for 500 days, 2. determine the distribution of transit times of each type of ship 3. determine the utilization of the three berths.

Code

```r
library(simmer)
library(simmer.plot)
library(ggplot2)
```

variables and Config.

```r
# setup
set.seed(42)

# create Simulation Envirenment
env <- simmer()
```

```r
SIM_TIME <- 24*500    # Simulation time in minutes
```

```r
ship_attrs <- function(){
  # get ship type as per 5:3:2=small:medium:big
  # determine ship type, 1 = small, 2= medium, 3 = big
  ship_type = sample(1:3,size=1, prob = c((5/10), (3/10), (2/10)))
  # randmly select berth to go
  # berth, 1 == berth1, 2 == berth2, 3 == berth3
  berth = "berth3"
  quan = 0
  unload_time = 0
  load_time = 0
  a = c("berth1", "berth2", "berth3")
  b = c("berth2", "berth3")

  if (ship_type == 1 ){
        berth = sample(a, size=1)
        quan = 1
        unload_time = rexp(1, 15)
        load_time = runif(1, 18, 30)
  }
    if (ship_type == 2 ){
        berth = sample(a, size=1)
        quan = 2
        unload_time = rexp(1, 30)
        load_time = runif(1, 26, 46)
    }
    if (ship_type == 3 ){
        berth = sample(b, size=1)
        quan = 4
        unload_time = rexp(1, 45)
        load_time = runif(1, 44, 68)
    }
  return(c(berth, quan, load_time, unload_time, ship_type))
}
```

```r
# This chunk is reponsible for crash
ship <- trajectory() %>%
  log_("arrives at the port") %>%
  set_attribute("dat", ship_attrs()) %>%
  log_(attr["dat"]) %>%
  set_prioritization(attr["dat"][5]) %>%

  seize(attr["dat"][1], attr["dat"][2] ) %>%
  timeout(floor(attr["dat"][4])) %>%
  timeout(floor(attr["dat"][3])) %>%
  release(attr["dat"][1], attr["dat"][2]) %>%
  log_("leaves the port")
```

```r
env %>%
    # add resources
    # berth1 resource is 2 // 1 medium ship == 2 small
    add_resource("berth1", 2) %>%
    # berth2 resource is 4 // 1 big ship == 2 medium == 4 small
    add_resource("berth2", 4) %>%
    # berth3 resource is 4 // 1 big ship == 2 medium == 4 small
    add_resource("berth3", 4) %>%
    # add ship generator with exponetial distribution
    add_generator("ship", ship, function() rexp(1, 1/26) ) %>%
    # start the simulation
    run(SIM_TIME)
```

output

This code is resulting in a crash of R studio.

Plot Things

```
plot(env, what = "resources", metric = "usage", c("berth1", "berth2", "berth3"))
plot(env, what = "resources", metric = "utilization", c("berth1", "berth2", "berth3"))
plot(env, what = "arrivals", metric = "activity_time")
plot(env, what = "arrivals", metric = "waiting_time")
plot(env, what = "arrivals", metric = "flow_time")
```

Conclusion

In this assignment I used discrete-event simulation library simmer to create a simulation, I was unsucessfull in creating a full simulation as per problem statment but

I was able to create some other simple simulations which can be found on my github account (https://github.com/electron0zero/R-projects/tree/master/hospital_simulation)

Final Conclusion

In these Assignments we learned about simulation and created simulations to model real life problems. Simulations are extreamly usefull, They save us time and resources and provides us the stastical anaysis of probelm (problem model to be exact, if model is inaccurate the results will be wrong)

Other then these assignments I took some time to model few things and creted simulations which can be found on my GitHub account (https://github.com/electron0zero/R-projects)