Simulation in R: Displacement of a Turtle

Narain Ritish Andrea A.V Hurtado

30 March 2023

UNIVERSITé DE FRANCHE-COMTé

Overview

UNIVERSITÉ DE FRANCHE-COMTÉ

Introduction of the study topic

The simulation problem

- La tortue est posée au point $(0, 0)$.
- A l'étape 1, la tortue se déplace de $+u$ avec u tiré au hasard uniformément parmi $(0, 1), (1, 0), (0, -1), (-1, 0)$.
- A l'étape elle se redéplace par le même procédé aléatoire. On répète ce déplacement n fois.
- On note N_n le nombre de fois où la tortue revient à un point déjà visité dans le passé.
- Utilisez la méthode de monté Carlo pour avoir une idée graphique de la loi de N_n avec $n = 100, 1000$ et 10000.

The Turtle

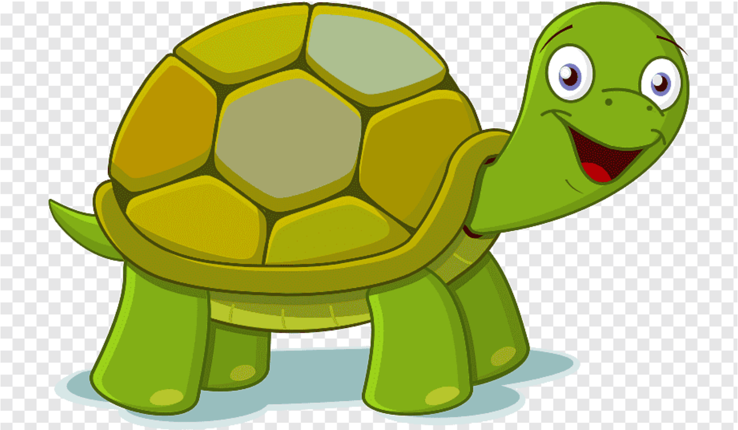


Figure: Turtle

1 Dimensional random walk algorithm

- Movements: $\{(x_0+1, y0), (x_0-1, y0), (x_0, y_0+1), (x_0, y_0-1)\}$
- Random movement with equal probabilities
- Movement in Z^2 with no. of simulations n.
- Multiple random walk with multiple simulations

1 D Random Walk Algorithm 1

```r
no_steps <- 1000
number_Walks <- 300

# We create a matrix to store the position of each step
positions <- matrix(0, ncol = no_steps + 1, nrow = number_Walks)

# We create a loop to calculate the position of each step

for (r in 1:number_Walks)
{
u <- 0 # The initial position at u= 0

for (i in 1:no_steps)
{
step <- runif(1, -1, 1) # Generate a uniform random number between -1 and 1
u <- u + step
positions[r, i + 1] <- u # The new position of 1 random walk
}
}
```

Figure: 1D Random Walk Algorithm 1

1-D Random Walk Algorithm 2

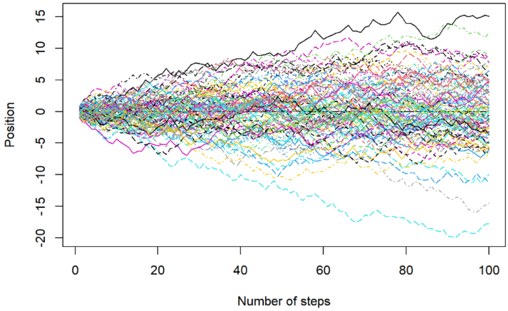
```
ran_walk_1D2 <-function(number_Walks,no_steps)
{
# We create a matrix of uniform generated variables U([-1,1])
mat_grid <- matrix(runif(number_Walks * no_steps, -1, 1), ncol = no_steps)

# We calculate the position of each steps here
positions2 <- apply(cbind(rep(0, number_Walks), mat_grid), MARGIN = 2, FUN = cumsum)

return(positions2)
}
```

Figure: 1-D Random walk algorithm 2

100 simulations 1-D with 100 Random Walk




```r
matplot(simul1d_1000, type = "l", col = 1:number_Walks, xlab = "Number of steps", ylab = "Position")
```

Figure: 100 random walks and simulations

Histogram: 100 simulations 1-D with 100 Random Walk

```
hist_100 <- hist(simul1d_100, breaks = 80, freq = FALSE,col = "dark green", fg = "dark green",
                 xlab = "Number of steps", ylab = "Length of random walk",
                  main = "Distribution of the random variables generated")
```

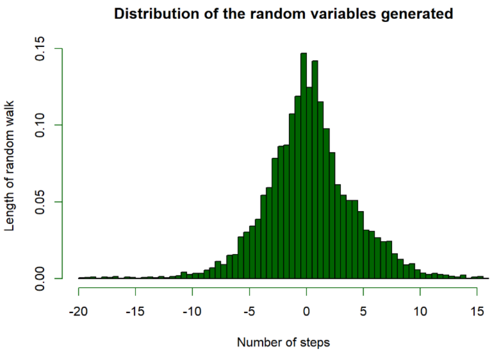
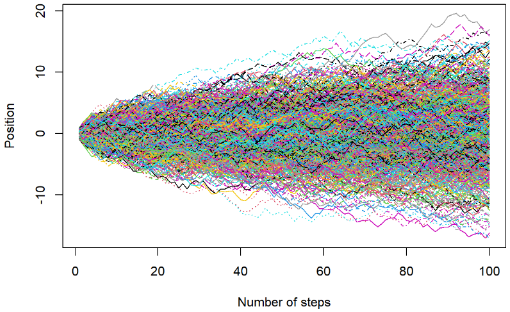


Figure: Histogram: 100 simulations

1000 simulations 1-D with 100 Random Walk




```
matplot(simul1d_10000, type = "l", col = 1:number_Walks, xlab = "Number of steps", ylab = "Position")
```

Figure: 1000 simulations

Histogram: 1000 simulations 1-D with 100 Random Walk

```
hist_1000 <- hist(simul1d_1000, breaks = 80, freq = FALSE,col = "dark green", fg = "dark green",
                  xlab = "Number of steps", ylab = "Length of random walk",
                   main = "Distribution of the random variables generated")
```

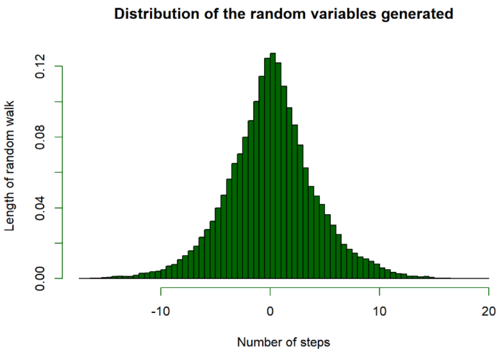


Figure: Histogram: 1000 simulations

10000 simulations 1-D with 100 Random Walk

```
matplot(simul1d_10000, type = "l", col = 1:number_Walks, xlab = "Number of steps", ylab = "Position")
```

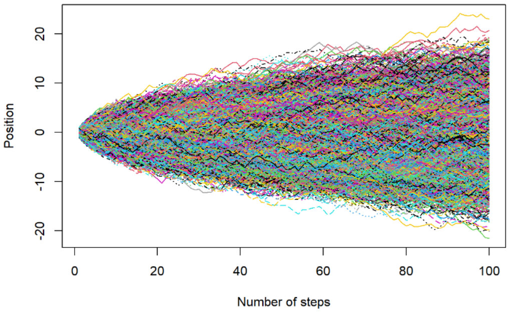
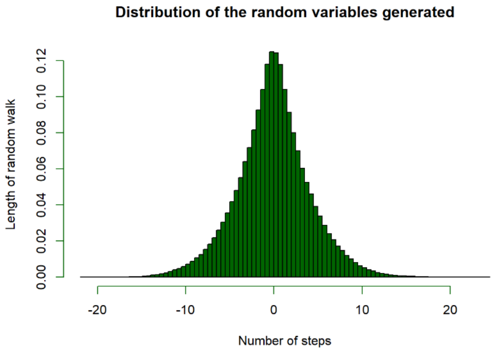


Figure: 10000 simulations

Histogram: 10000 simulations 1-D with 100 Random Walk

```r
hist_10000 <- hist(simul1d_10000, breaks = 80, freq = FALSE,col = "dark green", fg = "dark green",
           xlab = "Number of steps", ylab = "Length of random walk",
           main = "Distribution of the random variables generated")
```




```r
cdf_100 <- plot(ecdf(ran_walk_1D2(100,100)), main = "Empirical cumulative distribution with 100 steps ")
```

Figure: Histogram: 10000 simulations

2-D Random Walk Algorithm 1

```
Random_Walk_2D <- function(n_sim)

{
pos_desp <- c(-1, 0, 1) # possible displacement values

n_dup <-numeric(n_sim) # initialize a vector to store the number of duplicated positions in each simulation

for (i in 1:n_sim)
  {
 Alea <- sample(pos_desp, size = 2*n_sim, replace = TRUE) # generate a random sample of displacement values with replacement
 desp <- matrix(Alea, ncol = 2)                           # create a 2D matrix for the displacements with two columns, one f
or each                                                                     dimension
 pos <- apply(X = desp, MARGIN = 2, FUN = cumsum)         # apply the cumsum function to the columns to get step N+1

return(pos)

}
}
```

Figure: 2-D Random walk algorithm 1

Monte Carlo: Finding the distribution of the 2-D duplicate positions

- We will now estimate the displacements duplicated by the turtle by Monte Carlo.
- We select the chi-square and poisson distribution for comparison.
- N_n: number of duplicate positions
- n: Total number of simulations

2-D Random Walk Algorithm 2

```
set.seed(1234)
disp_2D_mc <- function(n_sim)

{

pos_desp <- c(-1, 0, 1) # possible displacement values

n_dup <-numeric(n_sim) # initialize a vector to store the number of duplicated positions in each simulation

for (i in 1:n_sim)
  {
 Alea <- sample(pos_desp, size = 2*n_sim, replace = TRUE) # generate a random sample of displacement values with replacement
 desp <- matrix(Alea, ncol = 2)                          # creates a 2D matrix for the displacements with two columns, one
for each                                                            dimension
 pos <- apply(X = desp, MARGIN = 2, FUN = cumsum)        # apply the cumsum function to the columns to get step N+1

 n_dup[i] <- sum(duplicated(pos))                        # Counts the sum of duplicated positions
}
prob_dup <- sum(n_dup>0)/n_sim                          # calculate the probability of duplicated positions (N_n/n)

Num_dup <-  max(n_dup)
hist_dup <- hist(n_dup, main = paste("Histogram of duplicated points (Simulations =", n_sim, ")"),
                                 xlab = "Number of duplicated points", ylab = "Frequency", breaks =n_sim/10,xlim = c(0,Num_
dup),col = "dark green", fg = "dark green", )

E_X <- mean(n_dup)
x <- seq(0, max(n_dup))
Poiss <- dpois(x, E_X) #Density of poisson distribution
Chi2 <- dchisq(x, E_X) #Density of chi-square distribution

lines(x,Poiss*sum(n_dup), col = "red")
lines(x,Chi2*sum(n_dup), col = "blue")
legend("bottomleft", legend = c("Simulated distribution", "Poisson distribution", "Chi Square distribution"),
       lty = c(1, 1), col = c("black", "red", "blue"), cex = 0.7)

return(n_dup)
print("Histogram of the Simulated duplicated positions\n", hist_dup,
      "\n Number of duplicated positions of the turtle\n", Num_dup,
      )
```

100 Simulation- Histogram

```r
#simul2d_100 <-disp_2D_mc(100)
simul2d_100.cdf <-plot(ecdf(disp_2D_mc(100)), main = "Empirical cumulative distribution 100 steps",col = "dark green", fg =
"dark green")
```

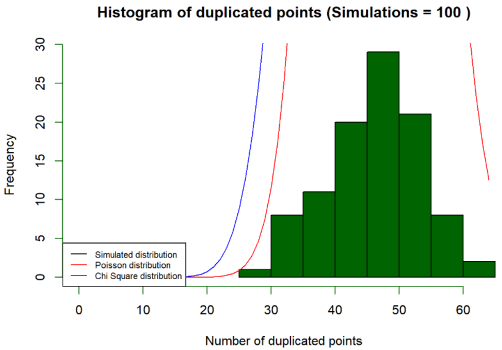


Figure: Histogram of 100 simulations

100 Simulation- Cumulative density function

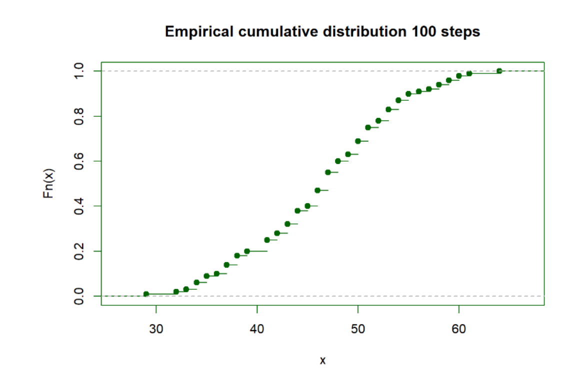


Figure: CDF for 100 simulations

1000 Simulation- Histogram

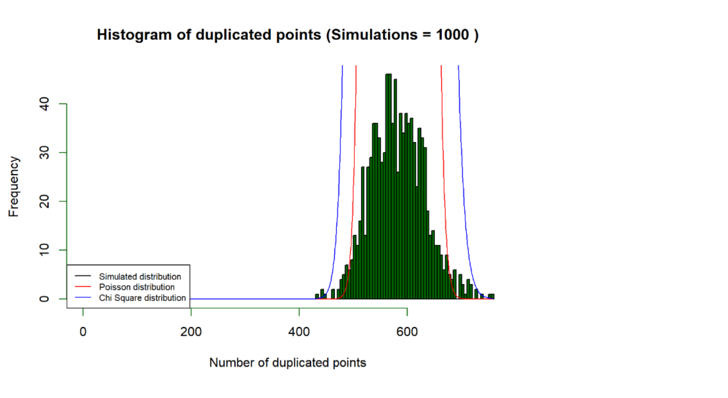


Figure: Histogram of 1000 simulations

1000 Simulation- Cumulative density function

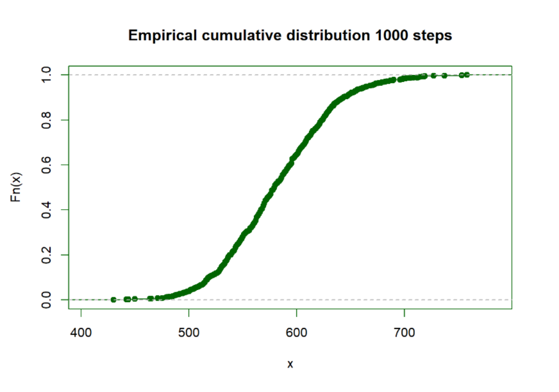


Figure: CDF for 1000 simulations

10000 Simulation- Histogram

```
#simul2d_10000 <-disp_2D_mc(10000)
simul2d_10000.cdf <-plot(ecdf(disp_2D_mc(10000)), main = "Empirical cumulative distribution 10000 steps" )
```

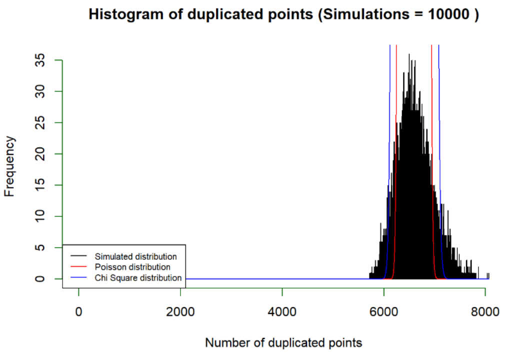


Figure: Histogram of 10000 simulations

10000 Simulation - Cumulative density function

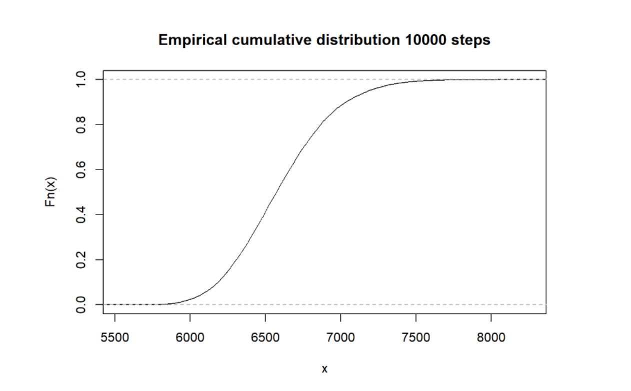


Figure: CDF for 10000 simulations

Conclusion

- Random Walk converges approximately to a normal distribution
- Duplicate positions of the random Walk converges approximately

to a poisson distribution

Questions?

Thank You ! Merci !