MTH209: HW 1

GROUP 9

January 2025

1 Estimate e using Probability theory

We know that the constant e is defined as the following limit:

$$e = \lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n$$

Method 1: Using Uniform Distribution

This method estimates e by leveraging the uniform distribution and the geometric mean.

Approach

1. **Generate Random Numbers:**
   - Generate n random numbers, $x_1, x_2, \ldots, x_n$, from a uniform distribution $U(0, 1)$.
2. **Calculate the Geometric Mean:**
   - Compute the geometric mean of the generated random numbers:

$$\text{Geometric Mean} = \left(\prod_{i=1}^{n} x_i\right)^{1/n}.$$

3. **Estimate e:**
   - Estimate e as the inverse of the geometric mean:

$$\hat{e} = \frac{1}{\text{Geometric Mean}}.$$

4. **Repeat for k Iterations:**
   - Repeat the above steps for k iterations, storing each estimate of e.
5. **Compute the Average Estimate:**

- Compute the average of the k estimates:

$$\text{Average Estimate} = \frac{1}{k}\sum_{i=1}^{k}\hat{e}_i.$$

6. **Analyze the Results:**

   - Observe how the estimates converge to the true value of $e \approx 2.718$ as k increases.

R Code Implementation

```r
e_estimate_2 = function(n, k) {
  estimates = numeric(k)    # Store estimates
  for (i in 1:k) {
    x = runif(n)            # Generate n random variables from U(0, 1)
    yn = prod(x)^(1/n)      # Geometric mean of x
    estimates[i] = 1 / yn   # Estimate of e
  }
  cumulative_avg = cumsum(estimates) / seq_len(k)  # Cumulative average
  return(cumulative_avg)    # Return cumulative averages
}

n = 100    # Number of random variables per iteration
k = 500    # Total number of iterations

# Generate estimates
cumulative_estimates = e_estimate_2(n, k)


# Plotting the results
plot(cumulative_estimates, type = "l", col = "blue", lwd = 2, ylim = c(2.6, 2.8),
     xlab = "Number of Iterations (k)", ylab = "Estimate of e",
     main = "Convergence of Estimated e")
abline(h = exp(1), col = "red", lwd = 2, lty = 2)  # Reference line for true value of e
legend("bottomright", legend = c("Estimated e", "True e"),
       col = c("blue", "red"), lwd = 2, lty = c(1, 2))
```

Analysis

This method estimates e by generating random numbers from a uniform distribution and calculates the geometric mean, with e approximated as the inverse of the mean. Over k iterations, the average of these estimates converges to 2.718.Initially, the values fluctuate, but they stabilize as k increases, demonstrating the **law of large numbers** . Using larger n improves the accuracy of individual estimates, while increasing k ensures better overall convergence. The method is simple and effective, though computationally intensive for larger values of n and k.

Conclusion

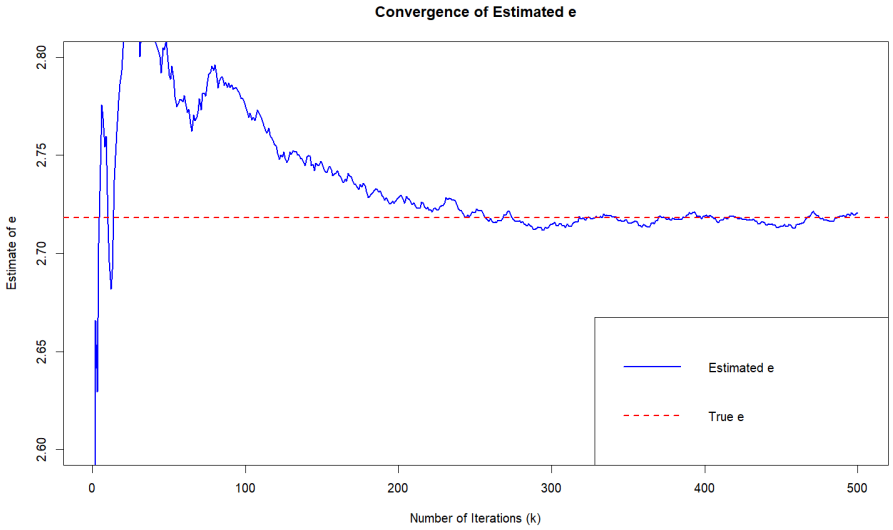

Figure 1: Visualization Of estimation of e.

This method effectively estimates e, with accuracy improving as n increases. This method provides more robust estimates at the cost of higher computational complexity.

2 Estimation of pi

A method for estimating π is by randomly sampling points in a square and checking how many fall within an inscribed circle. This method can be generalized to d dimensions by sampling points in a d-dimensional hypercube and checking if they fall within a hyper-sphere, estimating π accordingly.

Volume of a d-Dimensional Sphere

The volume V_d of a d-dimensional sphere of radius r is given by:

$$V_d = \frac{\pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)} r^d$$

Proof

To derive this formula, we consider the following steps:

Step 1: Volume as an integral The volume of a d-dimensional sphere is defined as the region in d-dimensional space where the Euclidean distance from the origin is less than or equal to r. This is expressed as:

$$V_d = \int_{||x|| \leq r} d^d x$$

Switching to spherical coordinates in d dimensions, the volume element becomes:

$$d^d x = r^{d-1}\, dr\, d\Omega_d$$

where r is the radial coordinate, and $d\Omega_d$ is the element of solid angle in d dimensions. The limits for r are $[0, r]$, and the integral over the solid angle gives the surface area of the unit sphere in $d-1$ dimensions, denoted S_{d-1}.

Step 2: Recursive relation for S_d The surface area of a d-dimensional sphere S_d is related to the volume of the d-dimensional ball by:

$$S_d = d \cdot V_d$$

This allows us to express the volume recursively:

$$V_d = \frac{S_d}{d}$$

Step 3: General formula for S_d The surface area of a unit sphere in d dimensions is:

$$S_d = \frac{2\pi^{d/2}}{\Gamma\left(\frac{d}{2}\right)}$$

Step 4: Combining results Substituting S_d into the recursive formula for V_d, we get:

$$V_d = \frac{1}{d} \cdot \frac{2\pi^{d/2}}{\Gamma\left(\frac{d}{2}\right)} r^d$$

Simplify to obtain:

$$V_d = \frac{\pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)} r^d$$

Conclusion Thus, the volume of a d-dimensional sphere of radius r is:

$$V_d = \frac{\pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)} r^d$$

Note : Here , since we are using Gamma function, we cannot use any odd no. of dimensions , because that will make the gamma output to be in terms of pi and we do not know pi as of yet. So we will run this for even dimensions i.e. 2, 4, 6, 8 and 10 here.

```
est_pi<-function(d,n){
  tot<-0
  inside<-0
  while(TRUE){
    point<-runif(d,min=-0.5,max=0.5)
    dist<-norm(point,"2")
    if(dist<0.5){
      inside<-inside+1


    }
    tot<-tot+1
    if(inside==n){
      break
    }

  }

  p<-inside/tot;                  #probabilty of point being inside
    the sphere
  pi_approx<-(gamma(d/2 + 1)*(2^d)*p)^(2/d)
  return(pi_approx)
}
est_pi(10,1e4)
points<-numeric(5)
for(i in 1:5){
  points[i]<-est_pi(2*i,1e4)
}


plot(x=c(2,4,6,8,10) ,y =points,ylim = c(3,3.2),type = "l",xlab = "
    Dimension",ylab="Estimated value of pi")
abline(h=pi,col = "red")
```

3 Test whether the generated observations are really "random" or not

3.1 Using Random Run Test

A run of a sequence is a maximal non-empty segment of the sequence consisting of adjacent equal elements. For example, the 22-element-long sequence

$$+ + + + - - - + + + - - + + + + + + - - - -$$

consists of 6 runs, with lengths 4, 3, 3, 2, 6, and 4. The run test is based on the null hypothesis that each element in the sequence is independently drawn from the same distribution.

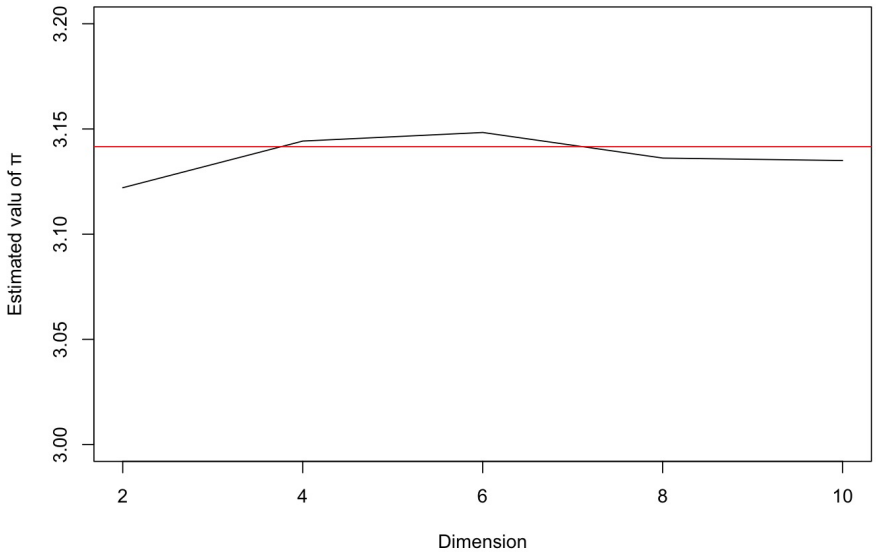

Figure 2: Visualization Of estimation of pi.

Under the null hypothesis, the number of runs in a sequence of N elements is a random variable whose conditional distribution given the observation of positive values and negative values

$$N = N_+ + N_-$$

is approximately normal, with :

$$\text{mean: } \mu = \frac{2N_+N_-}{N} + 1,$$

$$\text{variance: } \sigma^2 = \frac{2N_+N_-(2N_+N_- - N)}{N^2(N-1)} = \frac{(\mu-1)(\mu-2)}{N-1}.$$

Equivalently, the number of runs is

$$R = \frac{1}{2}\left(N_+ + N_- + 1 - \sum_{i=1}^{N-1} x_i x_{i+1}\right).$$

These parameters do not assume that the positive and negative elements have equal probabilities of occurring, but only assume that the elements are independent and identically distributed.

If the number of runs is significantly higher or lower than expected, the hypothesis of statistical independence of the elements may be rejected, this why, we here use only those runs which have no. of elements greater than 20.

```
library(tseries)
library(lawstat)
x=rnorm(100)
Check_whether_random<-function(x){
  y=numeric(length(x))

  x_median<-median(x)
  for(i in 1:length(x)){
    y[i]<-ifelse(x[i]<x_median,0,1)
  }
  result<-runs.test(y)

  p_value<-result$p.value
  if(p_value < 0.05     && !is.na(p_value) ){

    print("generated data is not random")
  }else{
    print("generated  data is random")
  }
  x_below_median<-x[x<= median(x)]
  x_above_median<-x[x>median(x)]
  if(length(x_below_median)>20){
    Check_whether_random(x_below_median)
  }
  if(length(x_above_median)>20){
    Check_whether_random(x_above_median)
  }

}
Check_whether_random(x)
```

The p-value is a statistical measure that helps us determine whether the results of your analysis are significant or not. It quantifies the strength of evidence against the null hypothesis in a hypothesis test. So, its value greater than 0.05 signifies the data is random.