CUNY DATA 609 HW4

Raphael Nash

9/11/2017

```
library("ggplot2")
```

P191 # 3

Using Monte Carlo simulation, write an algorithm to calculate an approximation to π by considering the number of random points selected inside the quarter x circle:

$$Q : x^2 + y^2 = 1, x \geq 0, y \geq 0$$

where the quarter circle is taken to be inside the square:

$$S : 0 \leq x \leq 1 \ \ and \ \ 0 \leq y \leq 1$$

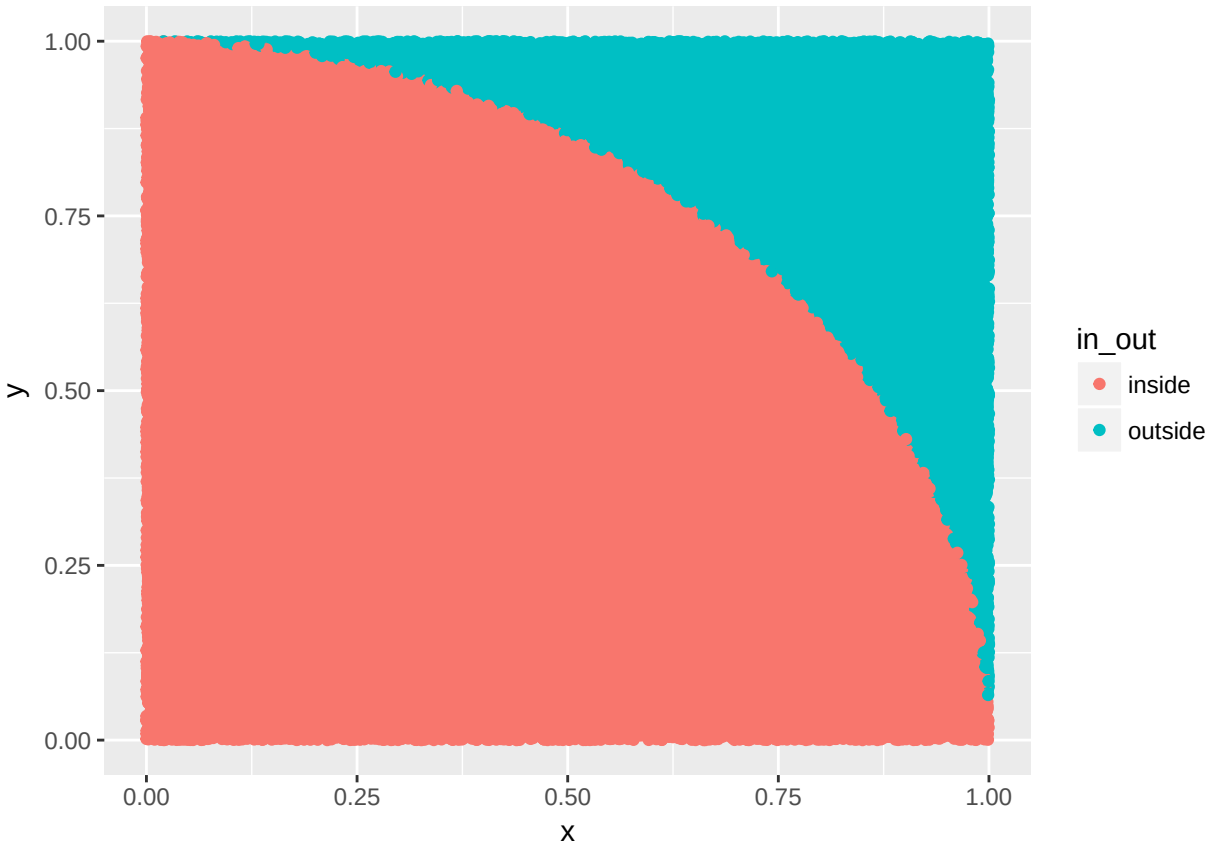
```
points_df <- data.frame (x=runif(100000), y= runif(100000) )

points_df$in_out <- NA

points_df$in_out <- ifelse(points_df$x^2 + points_df$y^2 < 1 , "inside" , "outside")

points_df$in_out <- factor(points_df$in_out)

ggplot(points_df, aes( x= x, y=y, colour = in_out )) + geom_point()
```



My estimate of π is: 3.1476

p194 # 1

```
mid_sq <- function(seed, n) {
  if (nchar( toString(seed))  < 4)  {
    print ( "enter an at least4 digit #")
    return()
  }

  v <-  vector()


  for ( i in 1:n) {

    seed <- seed^2

    seed_str <- toString(seed)

    while (nchar( seed_str ) < 8 )  {
      seed_str <- paste("0", seed_str, sep  = "")

    }

    seed <- as.numeric ( substring ( seed_str,2,6))
```

```
    v <- c(v,seed)
  }
  v

}
```

Use the middle-square method to generate:
a. 10 random numbers using $x_0 = 1009$.

```
mid_sq( 1009, 10)
```

```
##  [1] 10180  3632 31914  1850 34225 17135 93608 76245 81330 61456
```

b. 20 random numbers using $x_0 = 653217$.

```
mid_sq( 653217, 20)
```

```
##  [1] 26692 12462 55301  5820 38724 49954 49540 45421  6306 97656 53669
## [12] 88036 75033 62995 96837 37740 42430 80030 40480 63863
```

c. 15 random numbers using $x_0 = 3043$.

```
mid_sq( 3043, 15)
```

```
##  [1] 92598 57438 29912 94727 97320 47118 22010 84440 13011 69286 80054
## [12] 40864 66986 48712 37285
```

d. Comment about the results of each sequence. Was there cycling? Did each sequence degenerate rapidly? No I did not experience cycling. The number generated ran very fast.

p211 #2

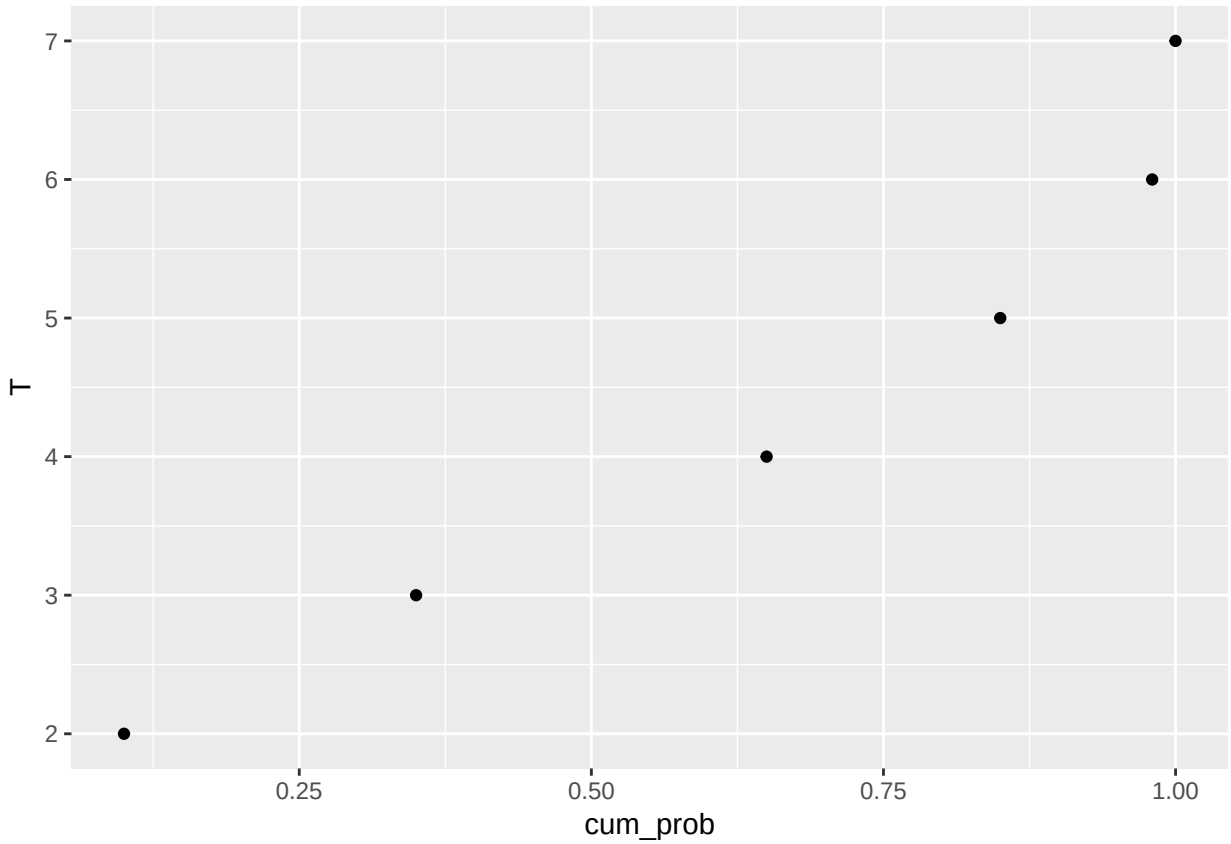
In many situations, the time T between deliveries and the order quantity Q is not xed. Instead, an order is placed for a speci c amount of gasoline. Depending on how many orders are placed in a given time interval, the time to ll an order varies. You have no reason to believe that the performance of the delivery operation will change. Therefore,you have examined records for the past 100 deliveries and found the following lag times, or extra days, required to ll your order:

```
delivery_df <- data.frame(
   T = c(2,3,4,5,6,7),
   num = c(10,25,30,20,13,2)
)
```

Construct a Monte Carlo simulation for the lag time submodel. If you have a handheld calculator or computer available, test your submodel by running 1000 trials and comparing the number of occurrences of the various lag times with the historical data.

```
delivery_df$observed_prob <- delivery_df$num / sum(delivery_df$num)
delivery_df$cum_prob <- cumsum(delivery_df$observed_prob)


ggplot(delivery_df, aes(y= T,x = cum_prob)) + geom_point()
```



This is the monte carlo simulation:

```
# the variables prev_prob and this_prob create the interval of cumulative probabilit to look in for the

sim_results <- vector()

pos <-0

for (i in runif(1000, min = 0 , max = 1 ))  {
  prev_prob <- 0


  for ( x in 1: NROW(delivery_df)){
    this_prob <- delivery_df[x,"cum_prob"]
    if ( (i >= prev_prob) && (i <= this_prob )   ) {
      sim_results <- c( sim_results, delivery_df[x,"T"]  )
      break
    }
    prev_prob <- this_prob

  }


}
```

```
sim_results <-    table(unlist(sim_results))
sim_results <- as.data.frame(sim_results)
```

```
sim_results$predicted_prob <- sim_results$Freq / sum(sim_results$Freq)
colnames (sim_results) <- c("T", "predicted_freq", "predicted_prob")

delivery_df <- merge(delivery_df, sim_results)
delivery_df
```

```
##   T num observed_prob cum_prob predicted_freq predicted_prob
## 1 2  10          0.10     0.10             95          0.095
## 2 3  25          0.25     0.35            244          0.244
## 3 4  30          0.30     0.65            273          0.273
## 4 5  20          0.20     0.85            208          0.208
## 5 6  13          0.13     0.98            156          0.156
## 6 7   2          0.02     1.00             24          0.024
```

Monte Carlo results....

Observed_prob is from the problem and predicted_prob is the results of the simulation.

```
delivery_df[c("T", "predicted_prob", "observed_prob")]
```

```
##   T predicted_prob observed_prob
## 1 2          0.095          0.10
## 2 3          0.244          0.25
## 3 4          0.273          0.30
## 4 5          0.208          0.20
## 5 6          0.156          0.13
## 6 7          0.024          0.02
```