Homework 2

Code:

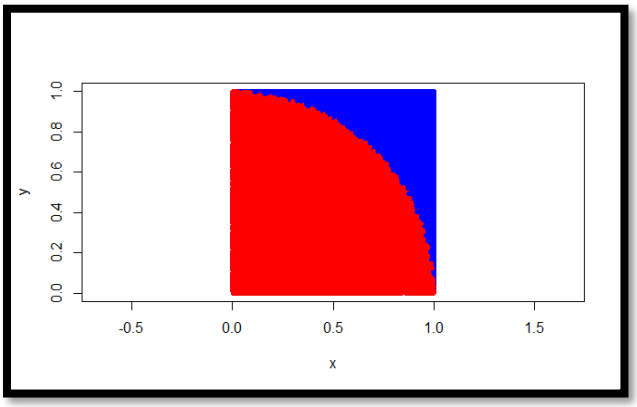
```r
# HW 2 Pi Estimation
# Inthat Sappipat 65011304

findpi <- function(n){ # create a funtion named findpi to find the value of pi of
                       # n coordinate (x,y) points
  set.seed(304) # set seed to ensure that the random numbers generated will be
                # the same each time the function is called with the same value of n
  x <- runif(n,min=0,max=1) # create a variable named x to store generated random number
                            # between 0 and 1
  y <- runif(n) # create a variable named y to store generated random number
  r <- sqrt((x^2)+(y^2)) # created a variable named r to store the distance of each point(x,y)
                         # from the origin(0,0)
  num.circle.dots <- sum(r <= 1) # create a variable name num.circle.dots to store
                                 # the count of coordinate (x,y) points inside the circle.
  num.square.dots <- n # create a variable name num.square.dots to store
                       # the total n coordinate (x,y) points
  ratio = num.circle.dots/num.square.dots # created a variable named ratio to store the ratio
                                          # of points inside the circle to the total number
                                          # of points generated
  my.pi <- 4 * ratio # created a variable named my.pi to store the Pi value that come from
                     # 4 multiply by the ratio
  different <- abs(my.pi - 3.14159) # created a variable named different to store the diffrence
                                    # between the estimated Pi value and default Pi value
  cat("The Value of pi:", my.pi, "\n") # show the output of estimated Pi value
  cat("The difference from 3.14159:", different, "\n") # show the output of the difference
                                                       # between the estimated Pi value
                                                       # and default Pi value
  plot(x, y, col=ifelse(r<=1,"red","blue"), asp=1, pch=20) # plot the graph
}

findpi(500000) # find estimated Pi value when input 500000 random coordinate (x,y) points
findpi(1000000) # find estimated Pi value when input 1000000 random coordinate (x,y) points
findpi(2000000) # find estimated Pi value when input 2000000 random coordinate (x,y) points
```

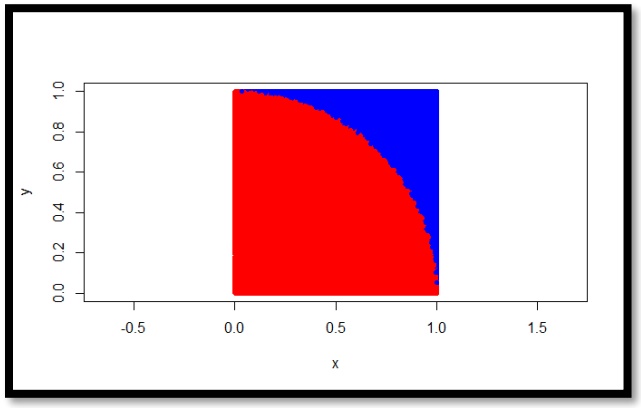
Result:

1.Findpi(500000):



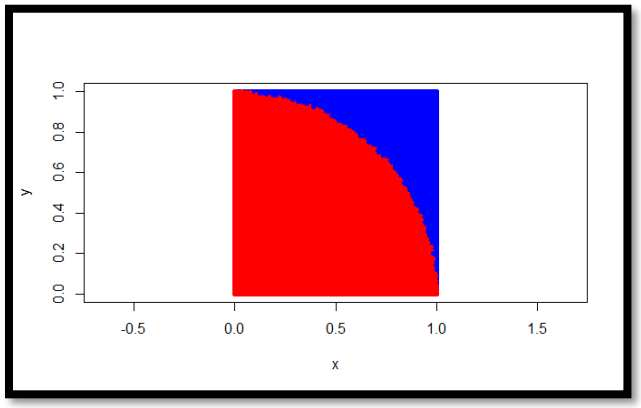

```
> findpi(500000)
The Value of pi: 3.14344
The difference from 3.14159: 0.00185
```

2.Findpi(1000000)




```
> findpi(1000000)
The Value of pi: 3.14384
The difference from 3.14159: 0.00225
```

3.Findpi(2000000)




```
> findpi(2000000)
The Value of pi: 3.14251
The difference from 3.14159: 0.00092
```

Result table:

Number of dots (n)	Pi value	Difference from 3.14159
500,000	3.14344	0.00185
1,000,000	3.14384	0.00225
2,000,000	3.14251	0.00092

Conclusion:

From the experiment, if we increase the number of dots, the Pi value seems to be closer to the classical Pi value (3.14159). In my opinion, increasing the number of random points will improve the accuracy and convergence of the estimation. Moreover, the ratio of the area of a quarter unit circle to the area of a unit square, which is $\pi / 4$, represents the relationship between the geometric properties of the circle and the square. I generate random points and compute ratios, allowing me to approximate the value of Pi.