Homework 4 W203

Natarajan Shankar

May 31, 2016

Homework 4 submission for 2016-0509 DATASCI W203,June 1, 2016

Homework 4a Question 1 - R Code

```
# prime.sieve
# A function to generate all prime numbers up to a user entered value
#
# Input : User entered integer, default entry is 100
#
# Output : A list of prime numbers up to the user entered number
#
prime.sieve <- function (n=100) {

  # Generate a list of all integers between 2 and the user enterend number n
  # 1 is not a prime number
  primeList <- seq(2, n)

  # Iterate over the list, cross out all prime number multiple up to n
  # test all multiples, go up to a n/i factor
  for (i in seq(2,sqrt(n))) {
    for (j in seq(2, n/i )) {

      # when multiple being tested is numericall bigger than the user entered number
      # skip over to the next
      if (i*j > n)
        break

      # Only cross out multiples of numbers that are not yet crossed out
      if (is.element(i*j, primeList))
        primeList <- primeList[which(primeList != i*j)]

    }
  }

  # Once complete, return the list of prime numbers
  return(primeList)
}
```

Home 4b Question 1 - Typical example

A typical example with a number 47

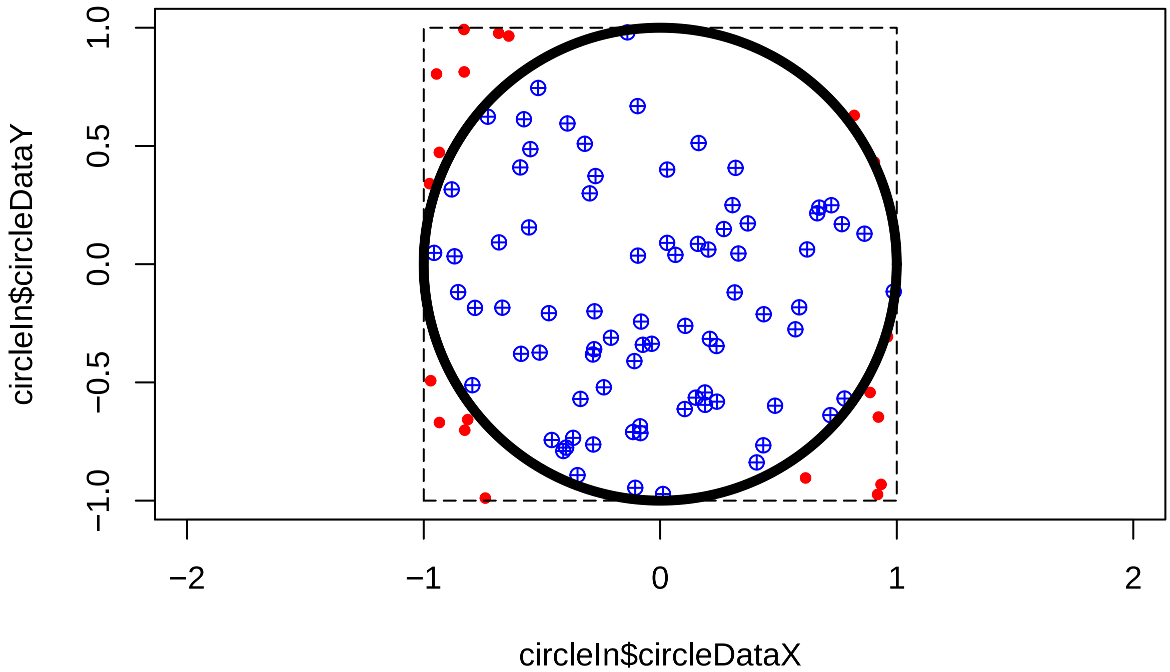
Prime numbers up to 47 are : 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47

Homework 4 submission for 2016-0509 DATASCI W203,June 1, 2016

Homework 4b Question 2 - R Code

```
# add libraries to be able to plot
library(plotrix)

# Pi is a well known value
pi = 3.14159

# Monte Carlo simulation to calculate Pi
# A function to generate an approximation for Pi
#
# Input : User entered number of (x,y) coordinates, default entry is 100
#
# Output : An estimate of the value of Pi
#
#pi.estimate <- function (n=100) {
    n=100
    set.seed(1212)

    # Generate x anf y coordinates in the range -1 to 1
    circleDataX <- runif(n, min=-1, max=1)
    circleDataY <- runif(n, min=-1, max=1)

    # add a flag to record whether a point is inside the circle (TRUE) or outside the circle (FALSE)
    inOrOut <- logical(n)

    # Create a data frame to maintain all the data
    circleDataFrame <- data.frame(circleDataX, circleDataY, inOrOut)

    # Sort the coordinates as to whether they are inside or on the circle or whether they are outside
    circleDataFrame$inOrOut <- ifelse(((circleDataFrame$circleDataX ^ 2) + (circleDataFrame$circleDataY

    # to make processing easier, subset the data frame into two
    # one for points that lie inside or on the circle
    # one for points that lie outside the circle
    circleIn <- subset(circleDataFrame, circleDataFrame$inOrOut == TRUE)
    circleOut <- subset(circleDataFrame, circleDataFrame$inOrOut == FALSE)

    # Plot infrastucture
    plot.new()
    frame()

    # Plot the inside points in Blue
    # Polt the outside points in Red
    #plot(circleDataX,circleDataY,asp=1,xlim=c(-1,1),ylim=c(-1,1))
    plot(circleIn$circleDataX,circleIn$circleDataY,asp=1,xlim=c(-1,1),ylim=c(-1,1), pch = 10, col = "blu
    points(circleOut$circleDataX, circleOut$circleDataY, pch = 20, col = "red")

    # Draw the circle
    draw.circle(0,0,1,nv=1000,border=NULL,col=NA,lty=1,lwd=5)
```

```r
    # Draw the square
    rect(-1,-1,1,1, lty=5)
```




```r
    # For convenience, compute the area of the square ( -1 to 1 side is of length 2)
    areaOfSquare <- 2 * 2

    # Formula for Pi estimation
    computedPi <- areaOfSquare * (nrow(circleIn) / n)

    # return the computed Pi value
    return(computedPi)
```

```
## [1] 3.2
```

```r
#}
```

Home 4b Question 1 - Typical example

A typical example with a number 100

For 100 (x, y) coordinates the pi estimate is : 3.2

Homework 4, Bonus Question Part 1

For a random variable X, the mean is:

$$m_x = E(\bar{X})$$

The Variance is:

$$\sigma^2 = E[X - \bar{X}]^2$$
$$\sigma^2 = \bar{X^2} - (\bar{X})^2$$

```
= E[X^2]  - (E[X])^2                **Equation 1**
```

If

$$C_i$$

is points inside circle

$$C_t$$

is total points plotted

If the probability p that a point lies inside the circle is 1 and if the probability that a point lies outside the circle is 0:

$$\frac{\pi}{4} = \frac{C_i}{C_t}$$

$$E[X] = (\frac{\pi}{4}) * 1 + (1 - \frac{\pi}{4}) * 0 = (\frac{\pi}{4})$$

$$E[X]^2 = (\frac{\pi}{4}) * (1^2) + (1 - \frac{\pi}{4}) * (0^2) = (\frac{\pi}{4})$$

Substituting back in Equation 1: E[X^2] - (E[X])^2 = $(\frac{\pi}{4}) - ((\frac{\pi}{4})^2)$ Which then is $(\frac{\pi}{4}) * (1 - (\frac{\pi}{4}))$
** Using pi value of 3.14159, Variance estimate for ONE TRIAL is 0.168548 **

or is the same as $\sigma^2 = (\frac{C_i}{C_t} * (1 - (\frac{C_i}{C_t}))$

The above is the variance estimate of $(\frac{\pi}{4})$

Hence, the Variance estimate of π is $\sigma^2 = (1/16) * (\frac{C_i}{C_t} * (1 - (\frac{C_i}{C_t}))$ ** Variance - Equation 2**

```
## Homework 4, Bonus Question Part 2 : Implement Equation 2 and do the Numerical procedure 1000 times ar

# Function variance.estimate implements Equation 2
variance.estimate <- function (n=100) {

    #
    # DO NOT USE A SEED, to allow results to vary
    #

    # Generate x and y coordinates in the range -1 to 1
    circleDataX <- runif(n, min=-1, max=1)
    circleDataY <- runif(n, min=-1, max=1)

    # add a flag to record whether a point is inside the circle (TRUE) or outside the circle (FALSE)
    inOrOut <- logical(n)

    # Create a data frame to maintain all the data
    circleDataFrame <- data.frame(circleDataX, circleDataY, inOrOut)

    # Sort the coordinates as to whether they are inside or on the circle or whether they are outside
    circleDataFrame$inOrOut <- ifelse(((circleDataFrame$circleDataX ^ 2) + (circleDataFrame$circleDataY

    # to make processing easier, subset the data frame into two
    # one for points that lie inside or on the circle
    # one for points that lie outside the circle
    circleIn <- subset(circleDataFrame, circleDataFrame$inOrOut == TRUE)
    circleOut <- subset(circleDataFrame, circleDataFrame$inOrOut == FALSE)
```

```r
    # Compute the Varianec Estimate by computing C-i/C-total
    varianceEstimate <- (((nrow(circleIn)/nrow(circleDataFrame))*(1-(nrow(circleIn)/nrow(circleDataFram

    # Return the variance estimate
    return(varianceEstimate)

}

# We are asked to do a 1000 iterations
# Vary the total number of plotted points randomly between 1000 and 10000
# compute the variance using equation 2 above

# Initialize 1000 entry vectors to hold data and results
varianceEstimate <- integer(1000)

# With each of 1000 tests, the number of plotted points is going to be changed. Keep track of total.
totalPoints <- integer(1000)

# Loop 1000 times over the variance.estimate() function
# start with count of 0
trials <- 1

#
# Main routine to collect 1000 variance estimates
#
while (trials <= 1000) {
  # Vary the total number of plotted points randomly between 1000 and 10000
  # One random integer between 1000 and 10000 is returned, save it
  totalPoints[trials] <- sample(1000:10000, 1)

  # Compute the variance estimate
  varianceEstimate[trials] <- (variance.estimate(totalPoints[trials]))

  #Increment iterations, need to stop at 1000
  trials <- trials + 1
}
```

Runtime example

Variance estimate with 1 trial 0

Variance estimate with 10 trials 0.005625

Variance estimate with 100 trials 0.0123188

Variance estimate with 1000 trials 0.0109328