IS609 Week 4 Homework

Ben Arancibia

September 19, 2015

191

3) Using the Monte Carlo simulation, write an algorithm to calculate an approximation to pi by considering the number of random points selected inside the quarter circle

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

where the quarter circle is taken to be inside the square

$$S : 0 \leq x \leq 1 \text{ and } 0 \leq y \leq 1$$

Use the equation $pi/4 =$ area $Q/areaS$.

```
qcircle<- function(x)
{
  val <- sqrt((-1 * x^2) + 1)
  return (val)
}
mcarlo <- function(n)
{
  dfpoints <- data.frame()
  x <- runif(n)
  y <- runif(n)
  yq <- qcircle(x)
  inside <- y <= yq

  dfpoints <- rbind(dfpoints, cbind(x, y, yq, inside))

  return (dfpoints)
}

mc <- 100
n_points <- 5000
dfmcpi <- data.frame()
for(i in 1:mc)
{
  dfmcpoints <- mcarlo(n_points)
  qca <- sum(dfmcpoints$inside) / nrow(dfmcpoints)
  qca
  pi_est <- qca * 4
  pi_est
  dfmcpi <- rbind(dfmcpi, cbind(i, pi_est))
}
head(dfmcpi)
```

```
##   i pi_est
## 1 1 3.1248
## 2 2 3.1080
## 3 3 3.1520
## 4 4 3.1640
## 5 5 3.1544
## 6 6 3.1920
```

```
median(pi_est)
```

```
## [1] 3.1592
```

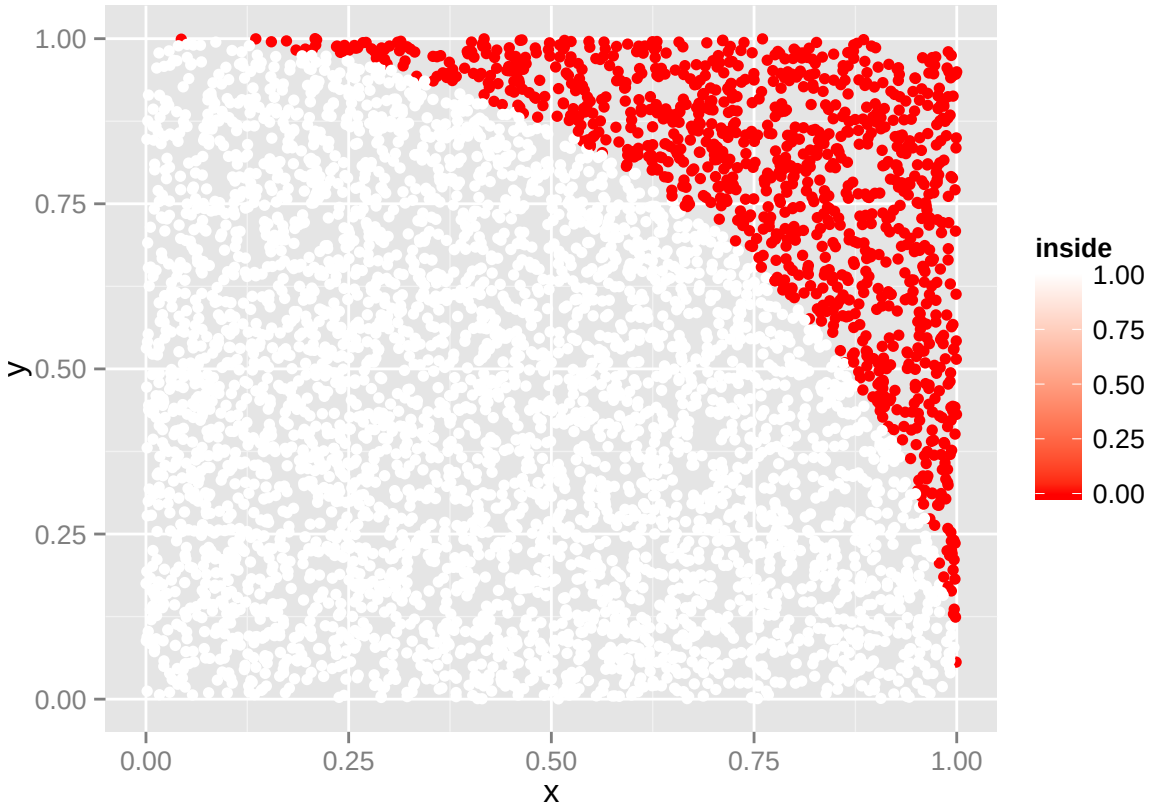
```
mean(pi_est)
```

```
## [1] 3.1592
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.1.3
```

```
f <- ggplot(data=dfmcpoints) + geom_point(aes(x=x, y=y, color=inside))+scale_colour_gradient(low="red",
f
```



194

1) Use the middle-square method to generate

```
middlesquare<- function(xn)
{
  l_x0 <- stringr::str_length(as.character(xn))
  s1 <- xn^2
  str1 <- as.character(s1)
  l <- stringr::str_length(str1)
  if(l < (l_x0 * 2))
  {
    str1 <- sprintf("%s%s", "0", str1)
  }
  start <- (l_x0 / 2) + 1
  end <- start + l_x0 - 1
  strRnd <- stringr::str_sub(str1, start, end)
  return (as.numeric(strRnd))
}
genRands <- function(n, seed)
{
  msRands <- c()
  xi <- seed
  for(i in 1:n)
  {
    xi <- middlesquare(xi)
    msRands[i] <- xi
  }
  return (msRands)
}
```

a) 10 random numbers using x0 = 1009

```
x0 <- 1009
genRands(10, x0)
```

```
##  [1] 180 324  49  40  60  60  60  60  60  60
```

b) 20 random numbers using x0 = 653217

```
x0 <- 653217
genRands(20, x0)
```

```
##  [1] 692449 485617 823870 761776 302674 611550 993402 847533 312186 460098
## [11] 690169 333248  54229  40784  63334  11195  25328  41507  22831  21254
```

c) 15 random numbers using x0 = 3043

```
x0 <- 3043
genRands(15, x0)
```

```
##  [1] 2598 7496 1900 6100 2100 4100 8100 6100 2100 4100 8100 6100 2100 4100
## [15] 8100
```

d) Comment about the results of each sequence. was there cycling? Did each sequence degenerate rapidly?

A degenerated to a constant after 4 iterations. B the values are decreasing but there have not cycled. C the values degenerated quickly. Each sequence did not generate rapidly.

NOTE I was not sure if the 5.3 homework problem was supposed to be in the project or problem set of the book. Based on previous homeworks our problems tend to be in the problem section of the book and not the projects section.

4) Given loaded dice according to the following distribution, use Monte Carlo simulation to simulate the sum of 300 rolls of two unfair dice.

```
##   roll pstart pfinish   p
## 1    1    0.0     0.1 0.1
## 2    2    0.1     0.2 0.1
## 3    3    0.2     0.4 0.2
## 4    4    0.4     0.7 0.3
## 5    5    0.7     0.9 0.2
## 6    6    0.9     1.0 0.1
```

```
##   roll pstart pfinish    p
## 1    1   0.00    0.30 0.30
## 2    2   0.30    0.40 0.10
## 3    3   0.40    0.60 0.20
## 4    4   0.60    0.70 0.10
## 5    5   0.70    0.75 0.05
## 6    6   0.75    1.00 0.25
```

```
dieoutcome <- function(dfDie, xk)
{
  res <- 0
  for(j in 1:nrow(dfDie))
  {
    if(dfDie$pstart[j] < xk && xk <= dfDie$pfinish[j])
    {
      res <- dfDie$roll[j]
    }
  }

  return(res)
}
# Loop through the 300 rolls of the Monte Carlo
# simulation to simulate the rolls of the unfair dice.
n = 300
rollcount <- data.frame(sum=seq(2,12), count=rep(0, 11))
for(i in 1:n)
{
  # 2 uniform random numbers.
  x <- runif(2)
  # Convert to die roll values
  x1Res <- dieoutcome(die1, x[1])
  x2Res <- dieoutcome(die2, x[2])
  # print(sprintf("Roll %d: %d, %d", i, x1Res, x2Res))
  # Increment the counter
  lookup <- rollcount$sum==x1Res+x2Res
  rollcount[lookup,]$count <- rollcount[lookup,]$count + 1
}
# Show the results.
```

```
rollcount$result <- rollcount$count / n
rollcount
```

```
##    sum count     result
## 1    2    11 0.03666667
## 2    3    14 0.04666667
## 3    4    33 0.11000000
## 4    5    45 0.15000000
## 5    6    40 0.13333333
## 6    7    45 0.15000000
## 7    8    32 0.10666667
## 8    9    34 0.11333333
## 9   10    24 0.08000000
## 10  11    16 0.05333333
## 11  12     6 0.02000000
```