MonteHall

Thiyanga Talagala

10/03/2020

```r
cal_fibonacci_sequence <- function(nterms){
# first two terms
n1 <-  0
n2 <- 1
count <- 2
# check if the number of terms is valid
if(nterms <= 0) {
print("Plese enter a positive integer")
} else {
    if(nterms == 1) {
        print("Fibonacci sequence:")
        print(n1)
    } else {
        print("Fibonacci sequence:")
        print(n1)
        print(n2)
        while(count < nterms) {
            nth = n1 + n2
            print(nth)
            # update values
            n1 = n2
            n2 = nth
            count = count + 1
          }
}
}
}

cal_fibonacci_sequence(7)
```

```
## [1] "Fibonacci sequence:"
## [1] 0
## [1] 1
## [1] 1
## [1] 2
## [1] 3
## [1] 5
## [1] 8
```

```r
print_sn <- function(){
  repeat{
    a <-  rnorm(1)
    if (a <= 1L){
      print(a)
    } else {
      break
    }

  }

}
```

```
print_sn <- function(){
  repeat{
    a <-  rnorm(1)
    if (a <= 1L){
      if(a <=0L){
      next
      } else {
        print(a)
      }
    } else {
      break
    }

  }

}
```

```
reveal_host_choice <- function(door) {
door.allocation <- sample(c("goat","goat","car"))
notchosen <- c(1:3)[-door]
if (door.allocation[door] == "goat") {
if (door.allocation[notchosen[1]] =="goat")
host_choice = paste(door.allocation[notchosen[1]],
paste("door", notchosen[1], sep=" "), sep="-")
else
host_choice = paste(door.allocation[notchosen[2]],
paste("door", notchosen[2], sep=" "), sep="-")
}
else {
d <- sample(notchosen, 1)
host_choice = paste(door.allocation[d], paste("door", d, sep=" "), sep="-")
}
return(host_choice)
}

reveal_host_choice(1)
```

```
[1] "goat-door 3"
```

```r
play_montehall <- function(door, strategy="switch") {
door.allocation <- sample(c("goat","goat","car"))
notchosen <- c(1:3)[-door]
host_choice <- sample(notchosen, 1) # player choice is car

if (door.allocation[door] == "goat") { # player chice is goat
host_choice <- ifelse (door.allocation[notchosen[1]] == "goat",
notchosen[1], notchosen[2])
}

# Now employ strategy
final_door <- door.allocation[door] # Strategy is to stay with original choice

if (strategy == "switch") {
d <- notchosen[which(notchosen != host_choice)]
final_door <- door.allocation[d]
}

final_door
}


play_montehall(1)
```

```
[1] "car"
```

```r
ncar.switch <- 0
for (i in 1:1000) {
if (play_montehall(door=1, strategy="switch") == "car")
ncar.switch <- ncar.switch+1
}
ncar.switch/1000
```

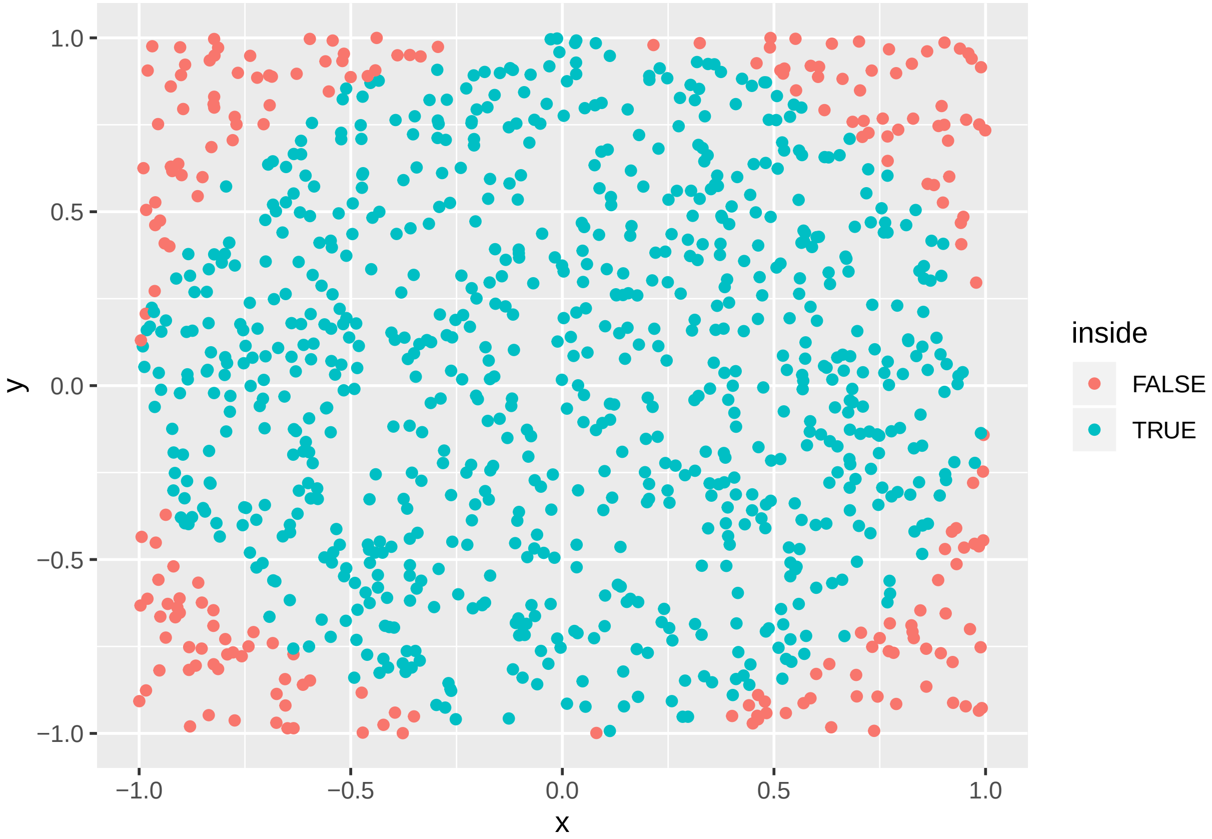
```
## [1] 0.653
```

```r
ncar.stay <- 0
for (i in 1:1000) {
if (play_montehall(door=1, strategy="stay") == "car")
ncar.stay <- ncar.stay+1
}
ncar.stay/1000
```

```
## [1] 0.327
```

```
library(ggplot2)
estimate_pi <- function(N, R){
x <- runif(N, min= -R, max= R)
y <- runif(N, min= -R, max= R)
is.inside <- (x^2 + y^2) <= R^2
pi.estimate <- 4 * sum(is.inside) / N
pi.estimate
data.fr <- data.frame(x=x, y=y, inside=is.inside)
qplot(data=data.fr, x=x, y=y, col=inside)
}

estimate_pi(1000, 1)
```



```
#estimate_pi(1000000, 1)
```

Bisection Method

```
cal_score_function <- function(x) {
  x * (x + 2) - 1
}

estimate_theta <- function(a, b, tol){
  fa <- cal_score_function(a)
  fb <- cal_score_function(b)
```

```r
  if ((fa * fb) > 0.0) {
  print("Function has same signs at ends of interval")
  } else {

    while (abs(a - b) > tol) {

    middle = (a + b) / 2.0
    print(paste("X: ", middle))

    if ((fa * fb) < 0.0) {
      b= middle
    } else {
      a = middle
    }

  }

  }
}

estimate_theta(0, 2, 0.0001)
```

```
[1] "X:  1"
[1] "X:  0.5"
[1] "X:  0.25"
[1] "X:  0.125"
[1] "X:  0.0625"
[1] "X:  0.03125"
[1] "X:  0.015625"
[1] "X:  0.0078125"
[1] "X:  0.00390625"
[1] "X:  0.001953125"
[1] "X:  0.0009765625"
[1] "X:  0.00048828125"
[1] "X:  0.000244140625"
[1] "X:  0.0001220703125"
[1] "X:  6.103515625e-05"
```