Week 15: Monte Carlo Methods

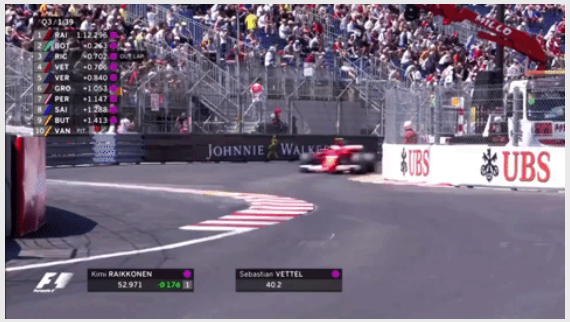
EMSE 6574 | John Paul Helveston | December 02, 2019

Monte Carlo, Monaco

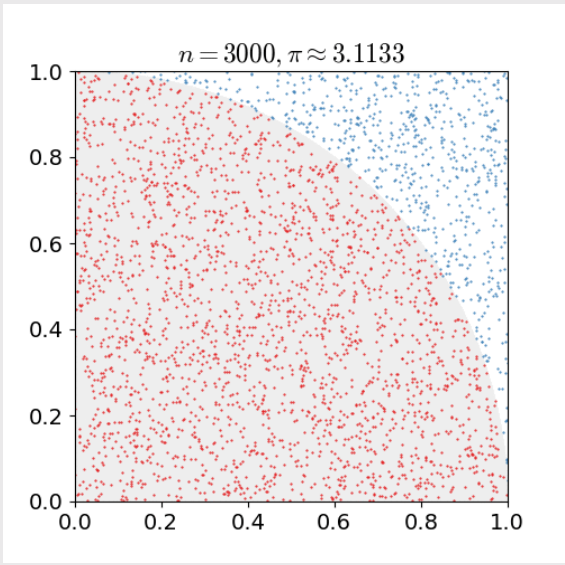
"Monte Carlo" is associated with 3 things

Gambling

Racing



Simulation



Monte Carlo Simulation

General process:

- Run a series of trials.
- In each trial, simulate an event (e.g. a coin toss, a dice roll, etc.).
- Count the number of "successful" trials

$$\frac{\#\ \text{Successful Trials}}{\#\ \text{Total Trials}} = \text{Observed Odds} \simeq \text{Expected Odds}$$

Law of large numbers: As # of trials increases, Observed Odds --> Expected Odds.

How would you measure if a coin is "fair"?

Run a series of trials and record outcome: "heads" or "tails"

```
coin <- c("heads", "tails")
N <- 10000
tosses <- sample(x = coin, size = N, replace = TRUE)
head(tosses)
```

```
## [1] "heads" "heads" "heads" "heads" "tails" "tails"
```

Probability of getting "heads":

```
sum(tosses == "heads") / N
```

```
## [1] 0.5016
```

Tossing a less-fair coin

Set the `prob` argument to a 40-60 coin

```
coin <- c("heads", "tails")
N <- 10000
tosses <- sample(x = coin, size = N, replace = TRUE, prob = c(0.4, 0.6))
head(tosses)
```

```
## [1] "tails" "heads" "heads" "tails" "heads" "heads"
```

Probability of getting "heads":

```
sum(tosses == "heads") / N
```

```
## [1] 0.3983
```

A more complex simulation: dice rolling

What is the probability of rolling one 6-sided dice 3 times and getting the sequence 1, 3, 5?

```
library(tidyverse)
dice <- c(1, 2, 3, 4, 5, 6)
N <- 10000
rolls <- tibble(
  roll1 = sample(x = dice, size = N, replace = TRUE),
  roll2 = sample(x = dice, size = N, replace = TRUE),
  roll3 = sample(x = dice, size = N, replace = TRUE)
)
```

```
dim(rolls)
```

```
## [1] 10000     3
```

```
head(rolls)
```

```
## # A tibble: 6 x 3
##   roll1 roll2 roll3
##   <dbl> <dbl> <dbl>
## 1     3     4     3
## 2     1     6     1
## 3     6     1     3
## 4     1     2     6
## 5     3     1     3
## 6     4     2     5
```

A more complex simulation: dice rolling

Simulated probability of getting sequence 1, 3, 5:

```
numObs <- rolls %>%
  filter(roll1 == 1, roll2 == 3, roll3 == 5) %>%
  nrow()
numObs / N
```

```
## [1] 0.0056
```

Actual probability of getting sequence 1, 3, 5:

```
(1/6)^3
```

```
## [1] 0.00462963
```

Practice

Use the `sample()` function and a monte carlo simulation to estimate the answers to these questions:

- If you flipped a coin 3 times in a row, what is the probability that you'll get three "tails" in a row?
- If you rolled 2 dice, what is the probability that you'll get "snake-eyes" (two 1's)?
- If you rolled 2 dice, what is the probability that you'll get an outcome that sums to 8?

Bonus:

What are the odds that five cards drawn from a 52-card deck will sum to a prime number? Set aces to 1 and all "face" cards (Jack, Queen, King) equal to 10. Hint: use `isPrime()` as a helper.

```
isPrime <- function(n) {
    if (n == 2) { return(TRUE) }
    for (i in seq(2, n-1)) {
        if (n %% i == 0) {
            return(FALSE)
        }
    }
    return(TRUE)
}
```

Discrete vs. continuous random numbers

Discrete

`sample()`: Takes random samples from `x`

Discrete vs. continuous random numbers

Discrete

`sample()`: Takes random samples from `x`

Example:

```
sample(x = c(0, 1), size = 10, replace = TRUE)
```

```
##  [1] 0 1 0 1 1 1 0 1 0 1
```

Discrete vs. continuous random numbers

Discrete

`sample()`: Takes random samples from `x`

Example:

```
sample(x = c(0, 1), size = 10, replace = TRUE)
```

```
##  [1] 0 1 0 1 1 1 0 1 0 1
```

Continuous

`runif()`: Takes random samples of numbers between upper and lower bound

Discrete vs. continuous random numbers

Discrete

`sample()`: Takes random samples from `x`

Example:

```
sample(x = c(0, 1), size = 10, replace = TRUE)
```

```
##  [1] 0 1 0 1 1 1 0 1 0 1
```

Continuous

`runif()`: Takes random samples of numbers between upper and lower bound

Example:

```
runif(n = 10, min = 0, max = 1)
```

```
##  [1] 0.4917393 0.7354521 0.8095845 0.8784901 0
##  [8] 0.9432800 0.1787074 0.8279663
```

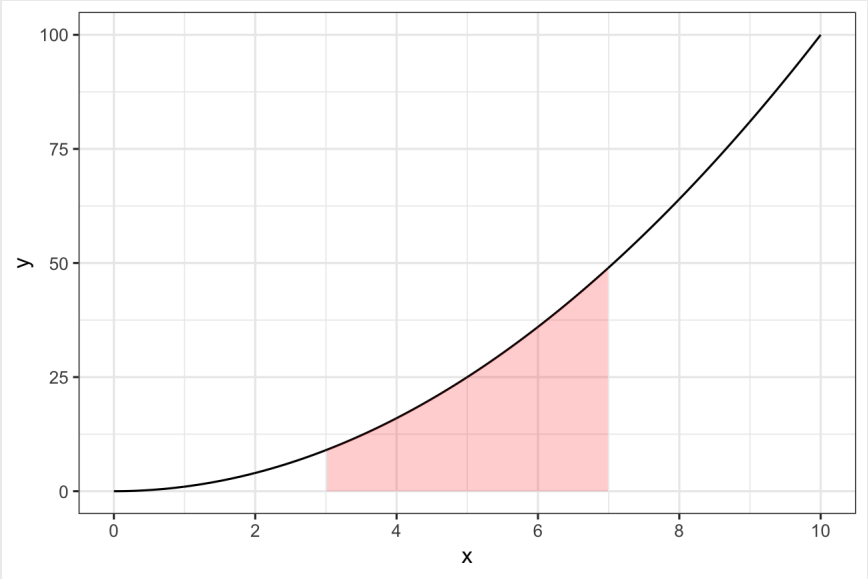
Monte Carlo Integration

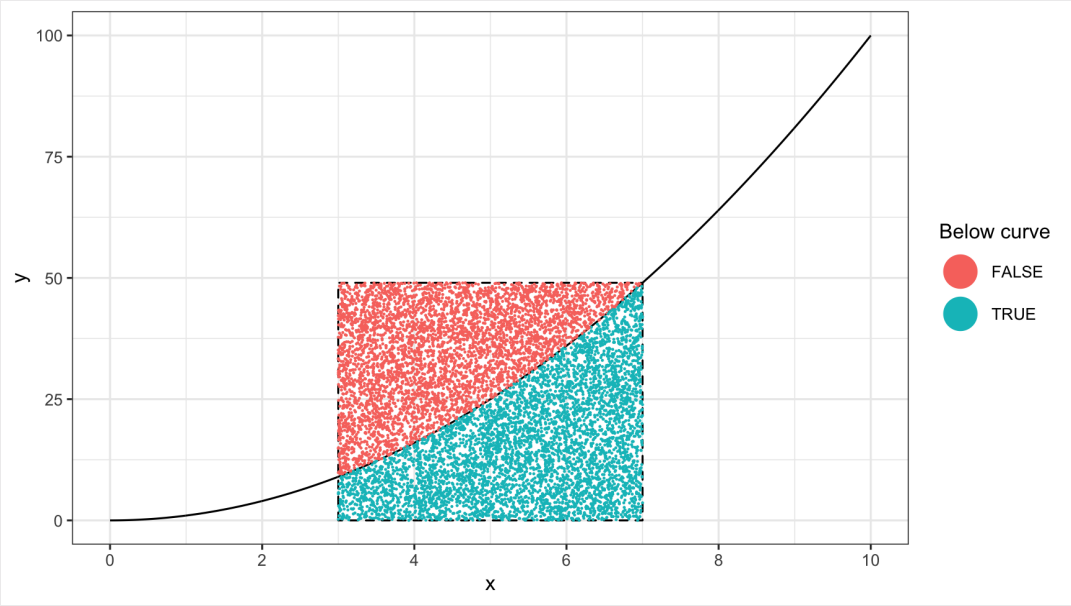
Integration = compute the area "under the curve"

Example:

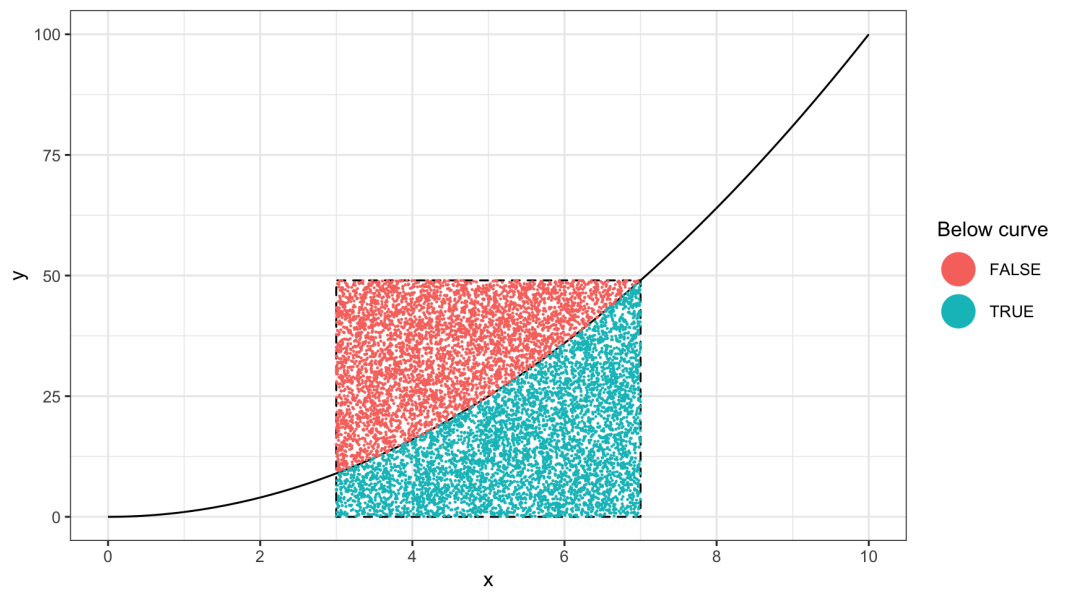
$$\frac{\#\text{ Points Under Curve}}{\#\text{ Total Points}} = \frac{\text{Area Under Curve}}{\text{Area of Rectangle}}$$

Find the area of $y = x^2$ between $3 < x < 7$





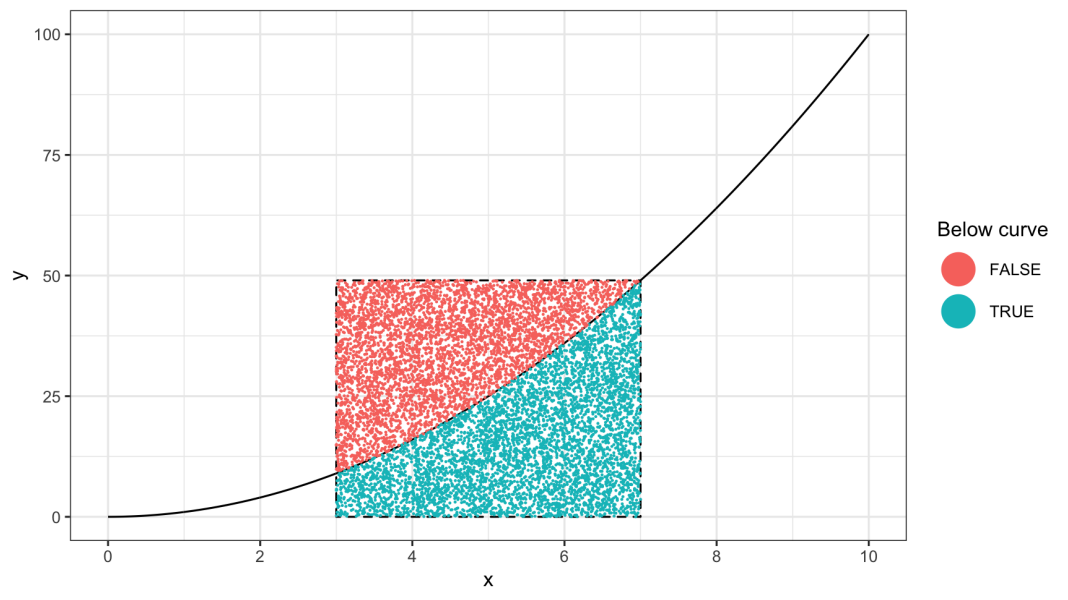
Monte Carlo Integration



$$\frac{\#\text{ Points Under Curve}}{\#\text{ Total Points}} = \frac{\text{Area Under Curve}}{\text{Area of Rectangle}}$$

$$\text{Area Under Curve} = \text{Area of Rectangle}\left(\frac{\#\text{ Points Under Curve}}{\#\text{ Total Points}}\right)$$

Monte Carlo Integration



Area of rectangle:

```
x <- 7 - 3
y <- 7^2 - 0
area_rectangle <- x*y
area_rectangle
```

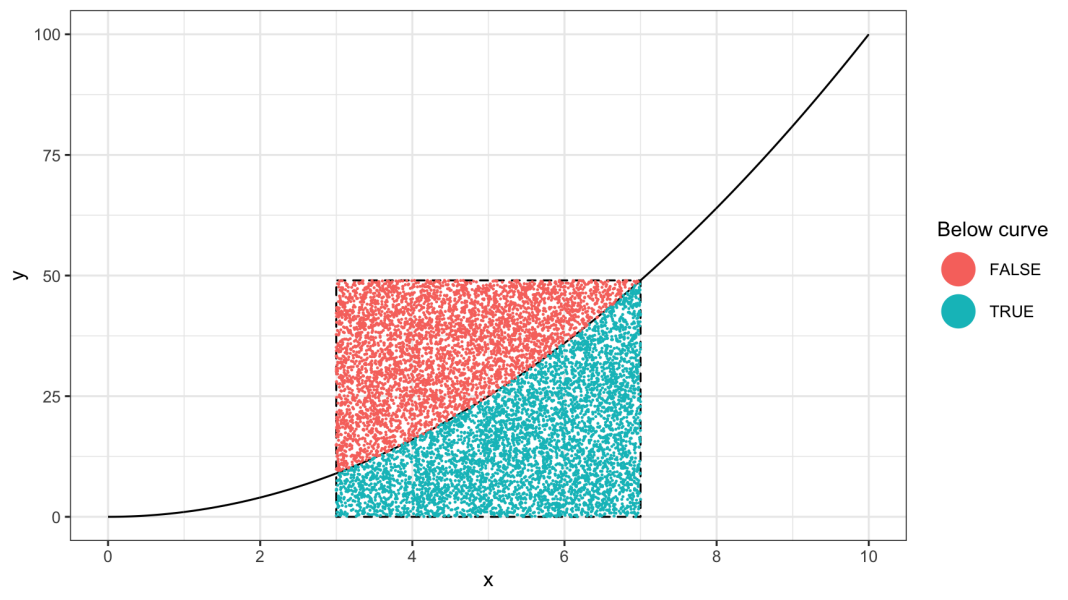
```
## [1] 196
```

Simulate points:

```
N <- 10000
points <- tibble(
    x = runif(N, min = 3, max = 7),
    y = runif(N, min = 0, max = 7^2)) %>%
    mutate(belowCurve = y < x^2)
head(points)
```

```
## # A tibble: 6 x 3
##       x     y belowCurve
##   <dbl> <dbl> <lgl>
## 1  6.22 12.2  TRUE
## 2  4.05 21.6  FALSE
## 3  5.67 28.8  TRUE
## 4  5.93 28.9  TRUE
## 5  6.34 41.0  FALSE
## 6  5.06  6.64 TRUE
```

Monte Carlo Integration



Area of rectangle:

```
x <- 7 - 3
y <- 7^2 - 0
area_rectangle <- x*y
area_rectangle
```

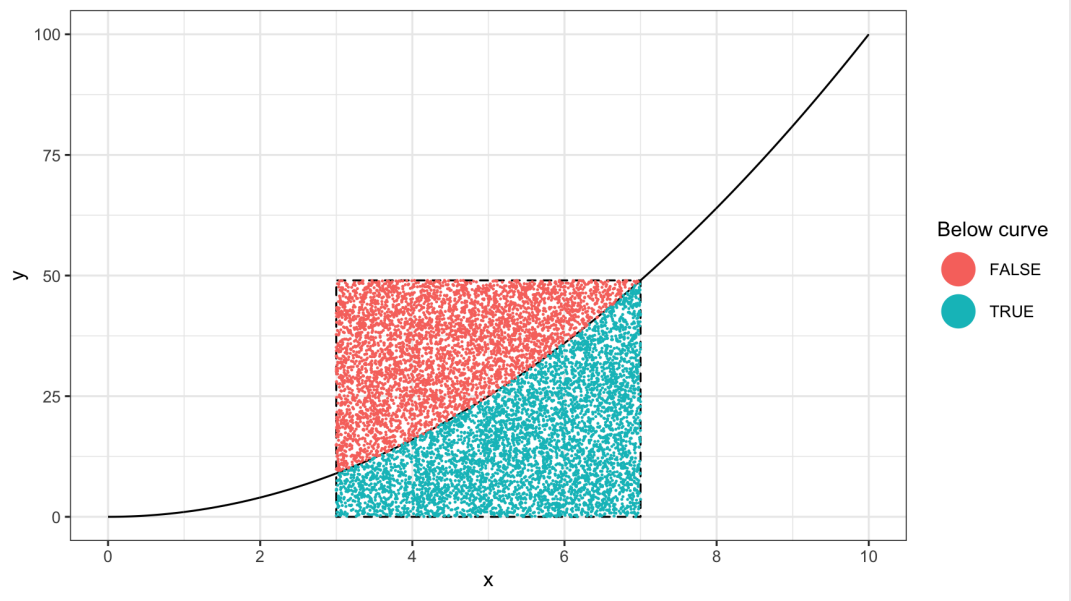
```
## [1] 196
```

Simulate points:

```
N <- 10000
points <- tibble(
    x = runif(N, min = 3, max = 7),
    y = runif(N, min = 0, max = 7^2)) %>%
    mutate(belowCurve = y < x^2)
sum(points$belowCurve)
```

```
## [1] 5330
```

Monte Carlo Integration



Area of rectangle:

```
x <- 7 - 3
y <- 7^2 - 0
area_rectangle <- x*y
area_rectangle
```

```
## [1] 196
```

Simulate points:

```
N <- 10000
points <- tibble(
    x = runif(N, min = 3, max = 7),
    y = runif(N, min = 0, max = 7^2)) %>%
    mutate(belowCurve = y < x^2)
sum(points$belowCurve)
```

```
## [1] 5330
```

Area under curve:

```
points_ratio <- sum(points$belowCurve) / N
areaUnderCurve <- area_rectangle * points_ratio
areaUnderCurve
```

```
## [1] 104.468
```

How did we do?

Simulated area under curve:

```
areaUnderCurve
```

```
## [1] 104.468
```

Actual area under curve:

$$\int_3^7 x^2 \mathrm{dx} = \left(\frac{x^3}{3}\right)\Big|_3^7 = \frac{7^3}{3} - \frac{3^3}{3} = 105.33\bar{3}$$

How did we do?

Simulated area under curve:

```
areaUnderCurve
```

```
## [1] 104.468
```

Actual area under curve:

$$\int_3^7 x^2 \mathrm{dx} = \left(\frac{x^3}{3}\right)\Big|_3^7 = \frac{7^3}{3} - \frac{3^3}{3} = 105.33\bar{3}$$

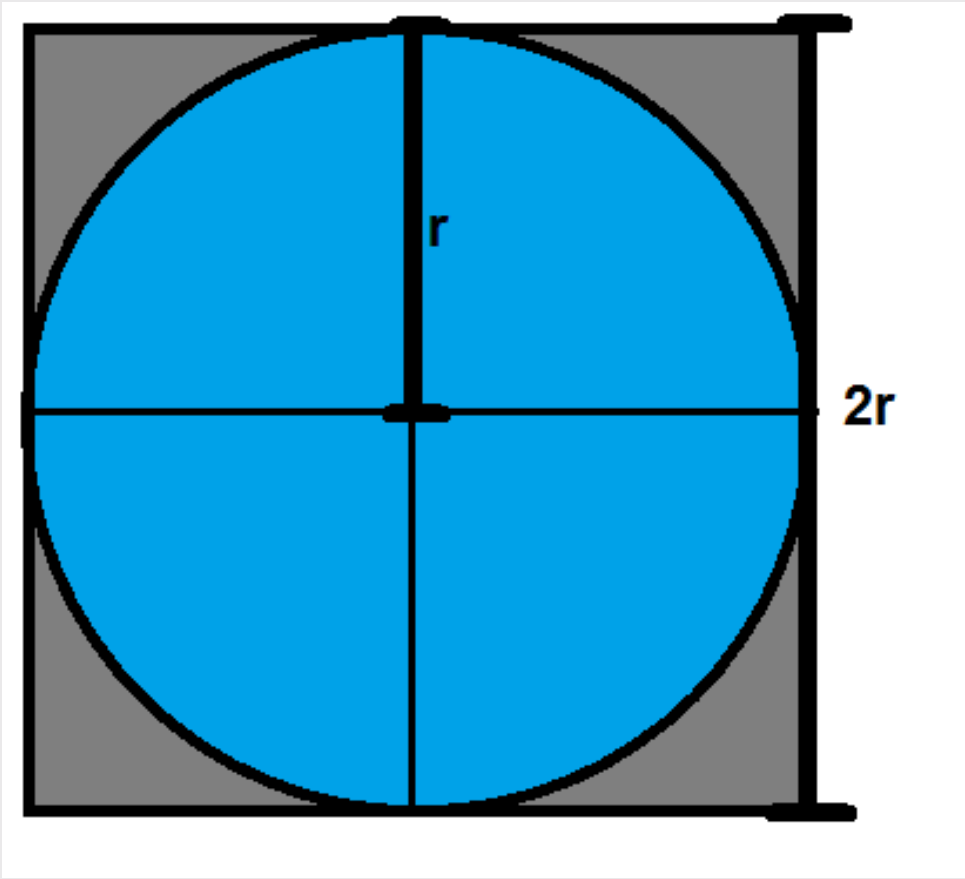
Error:

```
actualAreaUnderCurve <- (7^3 / 3) - (3^3 / 3)
error <- abs(actualAreaUnderCurve - areaUnderCurve)
error
```

```
## [1] 0.8653333
```

That's an error of 0.83 % - not bad!

Monte Carlo π



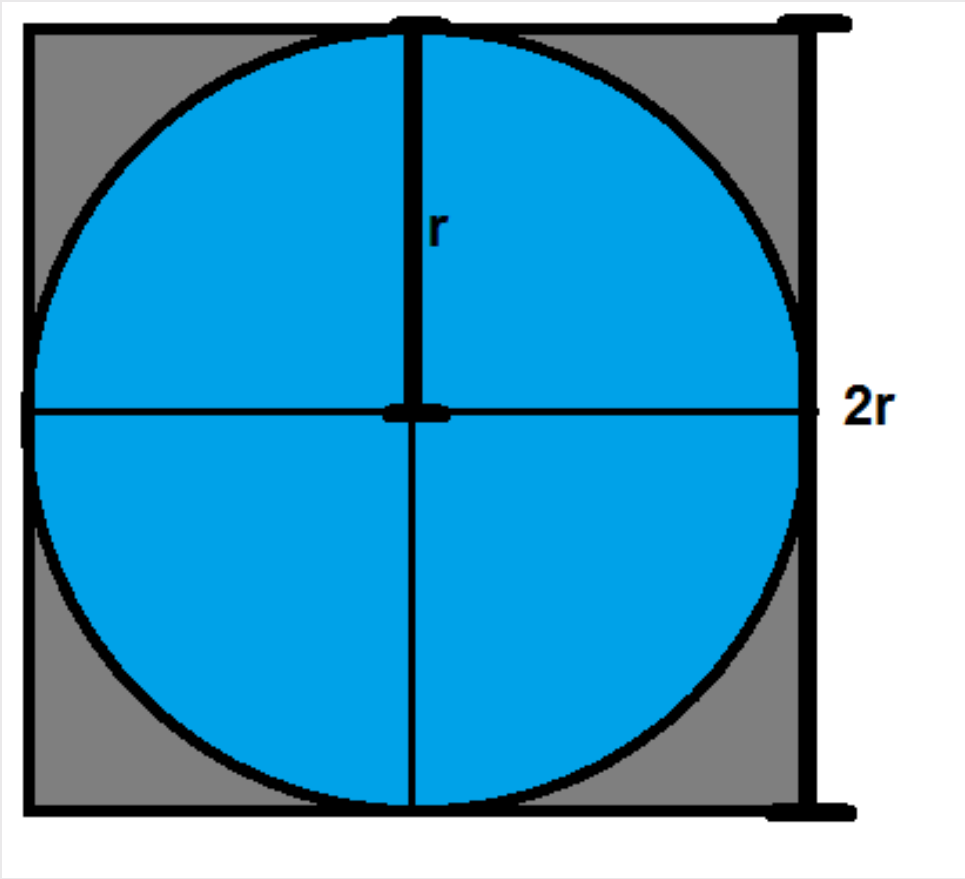
Area of a circle:

$$A_{circle} = \pi r^2$$

Area of square containing circle:

$$A_{square} = 4r^2$$

Monte Carlo π



Area of a circle:

$$A_{circle} = \pi r^2$$

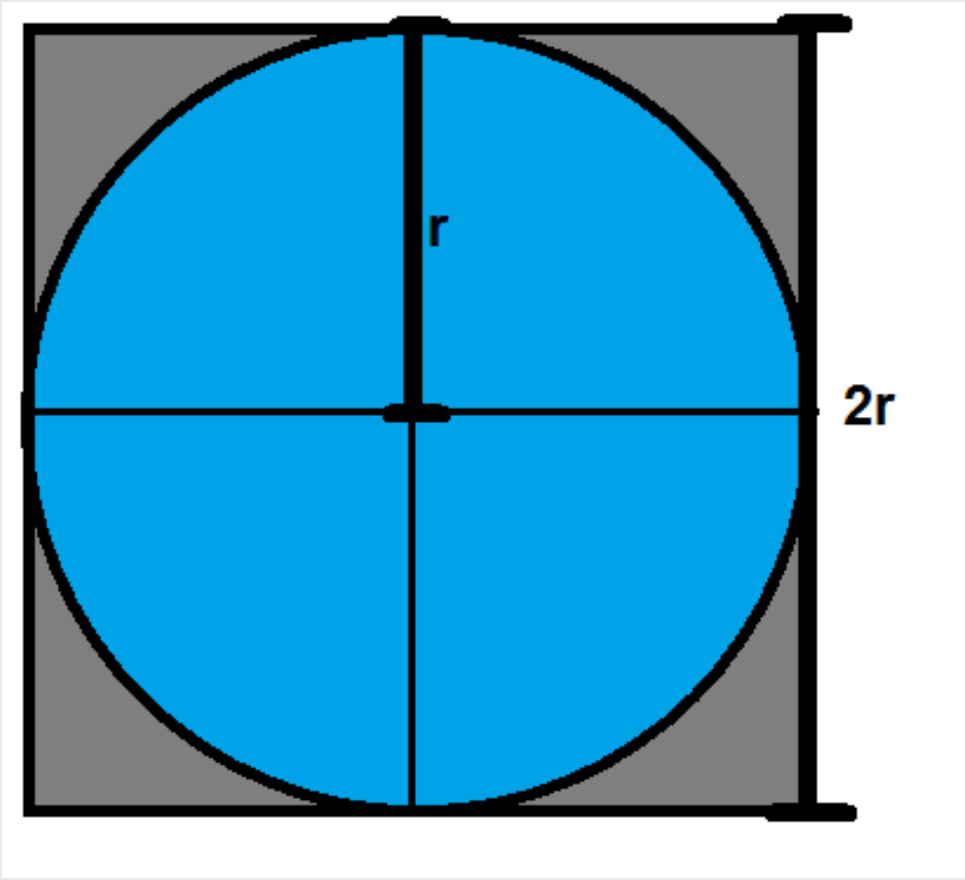
Area of square containing circle:

$$A_{square} = 4r^2$$

Ratio of areas = $\pi/4$:

$$\frac{A_{circle}}{A_{square}} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$$

Monte Carlo π



Area of a circle:

$$A_{circle} = \pi r^2$$

Area of square containing circle:

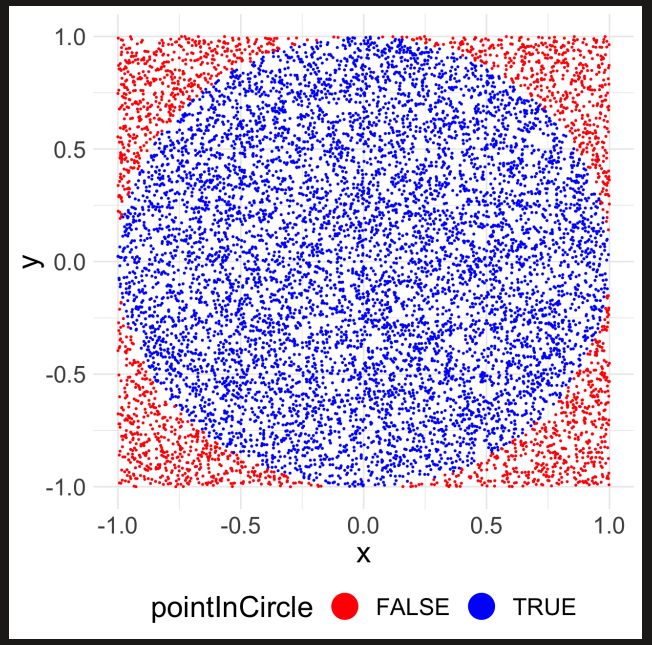
$$A_{square} = 4r^2$$

Ratio of areas = $\pi/4$:

$$\frac{A_{circle}}{A_{square}} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$$

$$\pi = 4\left(\frac{A_{circle}}{A_{square}}\right)$$

Practice: Estimate π

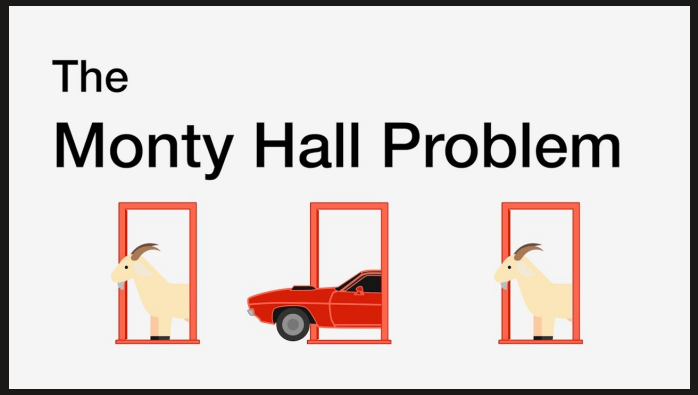


$$\pi = 4\left(\frac{A_{circle}}{A_{square}}\right)$$

1. Create a tibble with variables `x` and `y` that each contain 10,000 random points between -1 and 1, representing the (x, y) coordinates to a random point inside a square of side length 2 centered at `(x, y) = (0, 0)`. Hint: use `runif()`
2. Create a new variable, `radius`, that is equal to the distance from the center of the square (0, 0) to each (x, y) point.
3. Create the variable, `pointInCircle`, that is `TRUE` if the point lies *within* the circle inscribed in the square, and `FALSE` otherwise.
4. Create the scatterplot on the left (don't worry about the precise colors, dimensions, etc.).
5. Estimate π by multiplying 4 times the ratio of points inside the circle to the total number of points

The Monty Hall Problem

Practice: Monte Hall Problem



Choice 1 : Door 1, 2, or 3

Choice 2 : Swap doors, or keep your original?

In this simulation, the prize is always behind door #1:

- If you choose door #1, you must KEEP it to win.
- If you choose door #2 or #3, you must SWAP to win.

1. Create the tibble, `choices`, with two variables: `door` contains the first door chosen (`1`, `2`, or `3`); `swap` contains a logical value (`TRUE` or `FALSE`) for whether the contestant chose to swap door. Hint: use `sample()`
2. Create a new tibble, `wins`, which contains only the rows from `choices` that resulted in a win.
3. Compute the percentage of times the contestant won after swapping doors.

Reminders

1) Please fill the GW course feedback (see slack announcement)

2) Next week: final review

3) Final is Thursday, 12/12, from 10:20am - 12:20pm in Phillips 108