Week 13: *Monte Carlo Methods*

EMSE 4571: Intro to Programming for Analytics

John Paul Helveston

April 27, 2023

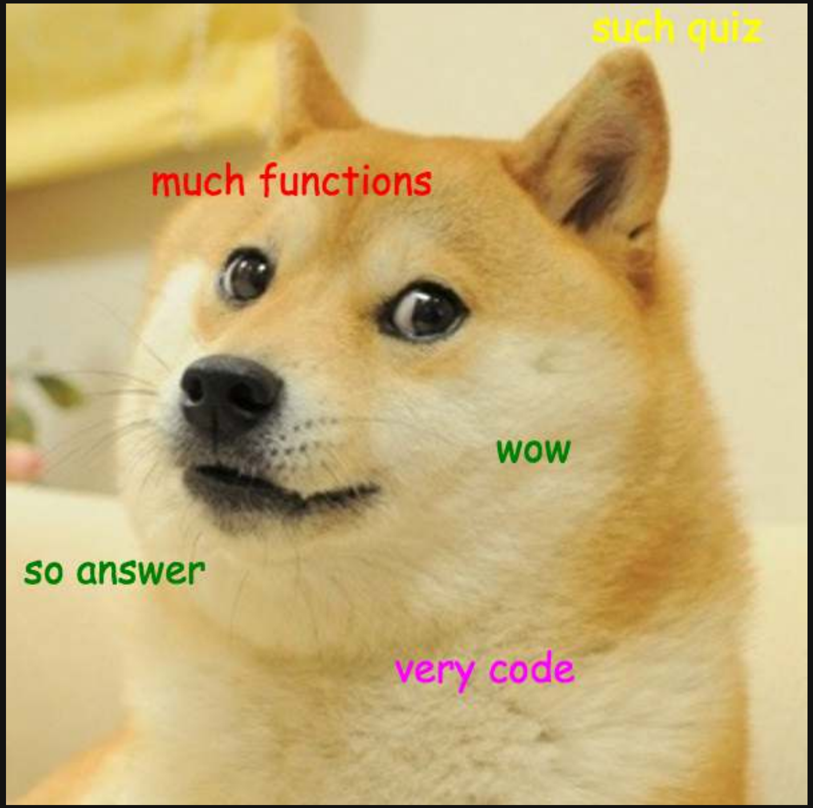
Quiz 7

10:00

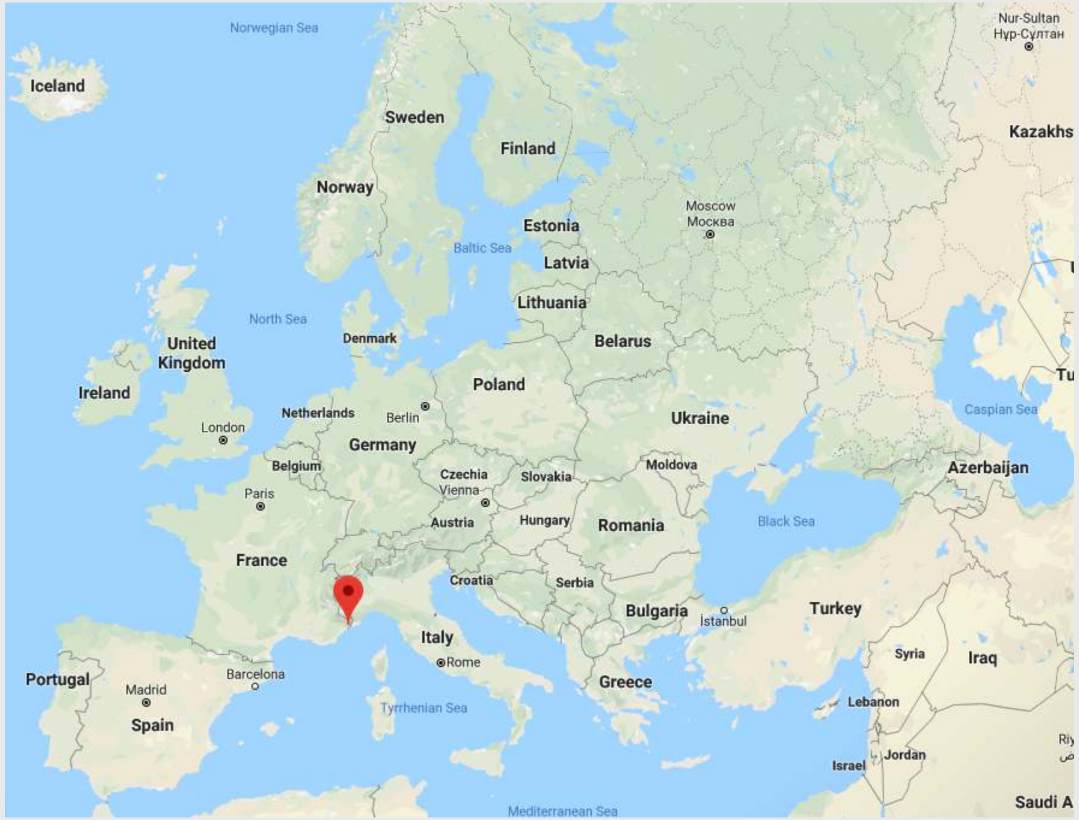
Write your name on the quiz!

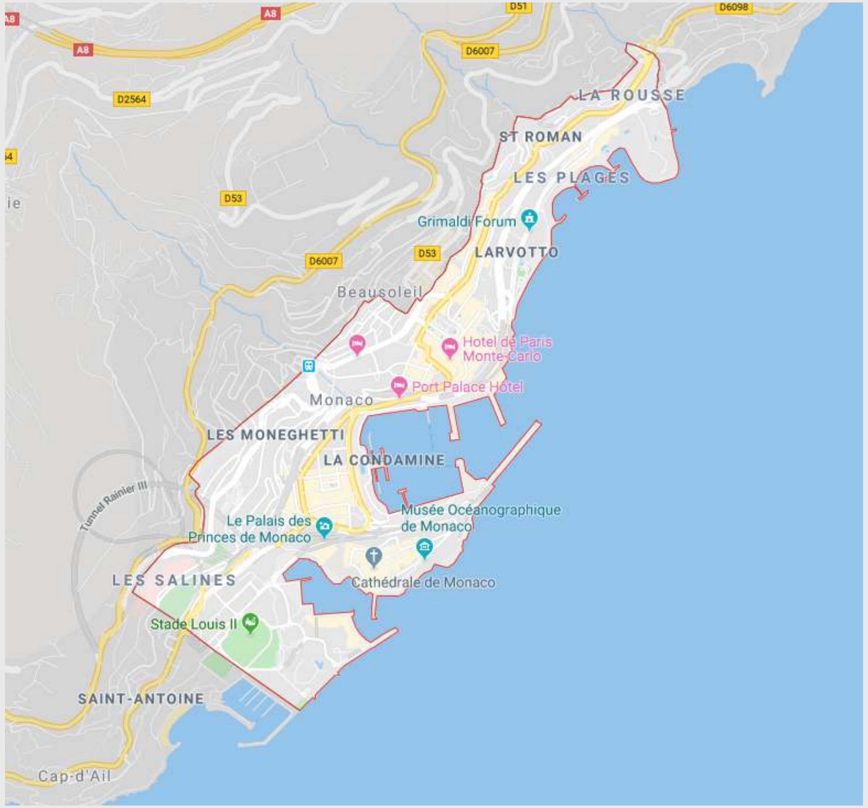
Rules:

- Work alone; no outside help of any kind is allowed.
- No calculators, no notes, no books, no computers, no phones.



Monte Carlo, Monaco



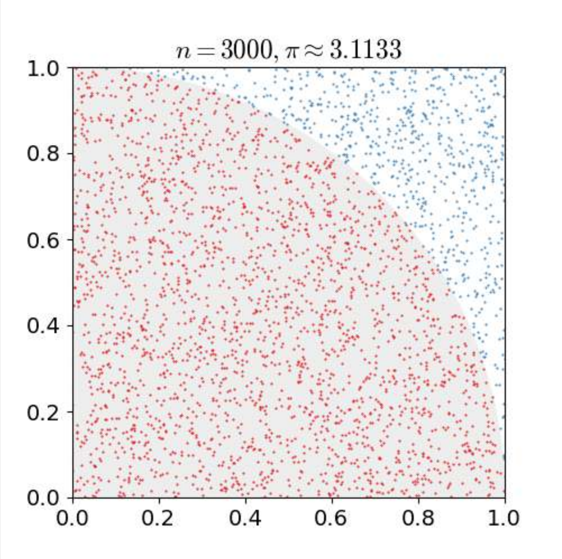


"Monte Carlo" is associated with 3 things

Gambling

Racing

Simulation



Week 13: *Monte Carlo Methods*

1. Monte Carlo Simulation

BREAK

2. Monte Carlo Integration

Week 13: *Monte Carlo Methods*

1. Monte Carlo Simulation

BREAK

2. Monte Carlo Integration

Monte Carlo Simulation: *Computing Probability*

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 *N* 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"

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

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

Probability of getting "heads":

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

```
#> [1] 0.5018
```

Tossing an unfair 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) # Preview first few tosses
```

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

Probability of getting "heads":

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

```
#> [1] 0.4026
```

A more complex simulation: *dice rolling*

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

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

```r
head(rolls)
```

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

A more complex simulation: *dice rolling*

Simulated probability of getting sequence 1, 3, 5:

```
successes <- rolls %>%
  filter(roll1 == 1 & roll2 == 3 & roll3 == 5)

nrow(successes) / N
```

```
#> [1] 0.0049
```

Actual probability of getting sequence 1, 3, 5:

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

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

Your Turn: Coins & Dice

15:00

Using the `sample()` function, conduct 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?

When `replace = FALSE`

Sometimes events cannot be independently simulated

What are the odds that 3 cards drawn from a 52-card deck will sum to 13?

- Aces = 1
- Jack = 10
- Queen = 10
- King = 10

When `replace = FALSE`

Sometimes events cannot be independently simulated

```
cards <- c(seq(1, 10), 10, 10, 10)
deck <- rep(cards, 4) # Rep because there are 4 suits
length(deck)
```

```
#> [1] 52
```

Draw 3 cards from the deck *without replacement*:

```
draw <- sample(x = deck, size = 3, replace = FALSE)
draw
```

```
#> [1]  9  4 10
```

When `replace = FALSE`

Note: You can't draw more than 52 cards *without replacement*:

```r
draw <- sample(x = deck, size = 53, replace = FALSE)
```

```
#> Error in sample.int(length(x), size, replace, prob): cannot take a sample larger than
the population when 'replace = FALSE'
```

When `replace = FALSE`

What are the odds that 3 cards drawn from a 52-card deck will sum to 13?

Repeat the 3-card draw *N* times:

```r
N <- 100000
count <- 0

for (i in 1:N) {
   draw <- sample(x = deck, size = 3, replace = FALSE)
   if (sum(draw) == 13) {
      count <- count + 1
   }
}

count / N # Compute the probability
```

```
#> [1] 0.03662
```

Your Turn: Cards

15:00

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

- What are the odds that four cards drawn from a 52-card deck will have the same suit?
- What are the odds that five cards drawn from a 52-card deck will sum to a prime number?
- Aces = 1
- Jack = 10
- Queen = 10
- King = 10

Hint: use `isPrime()` to help:

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

Break

05:00

Week 13: *Monte Carlo Methods*

1. Monte Carlo Simulation

BREAK

2. Monte Carlo Integration

Discrete vs. continuous random numbers

Discrete

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

```r
sample_discrete <- sample(
  x       = c("heads", "tails"),
  size    = 5,
  replace = TRUE
)

sample_discrete
```

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

Continuous

`runif()`
Takes random samples between bounds

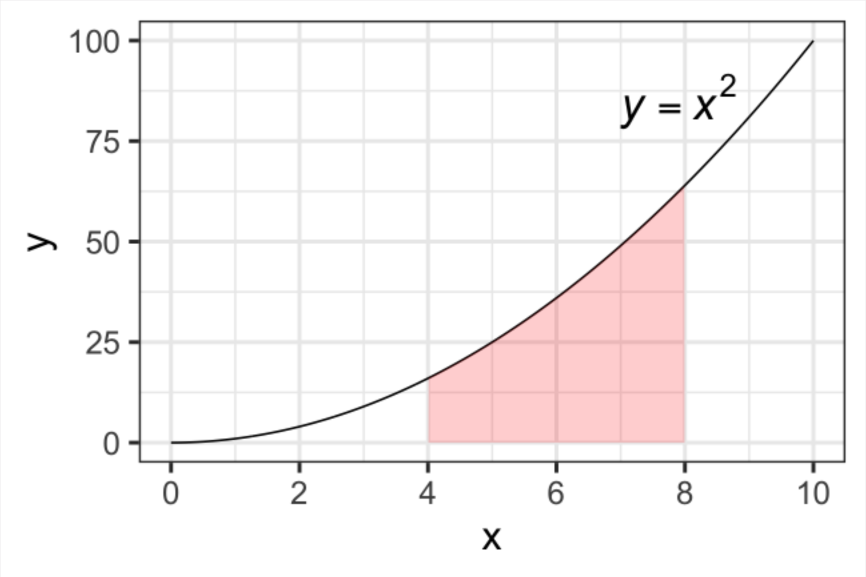
```r
sample_continuous <- runif(
  n   = 5,
  min = 0,
  max = 1
)

sample_continuous
```

```
#> [1] 0.7536088 0.2830033 0.6428447
0.4526490 0.7055930
```

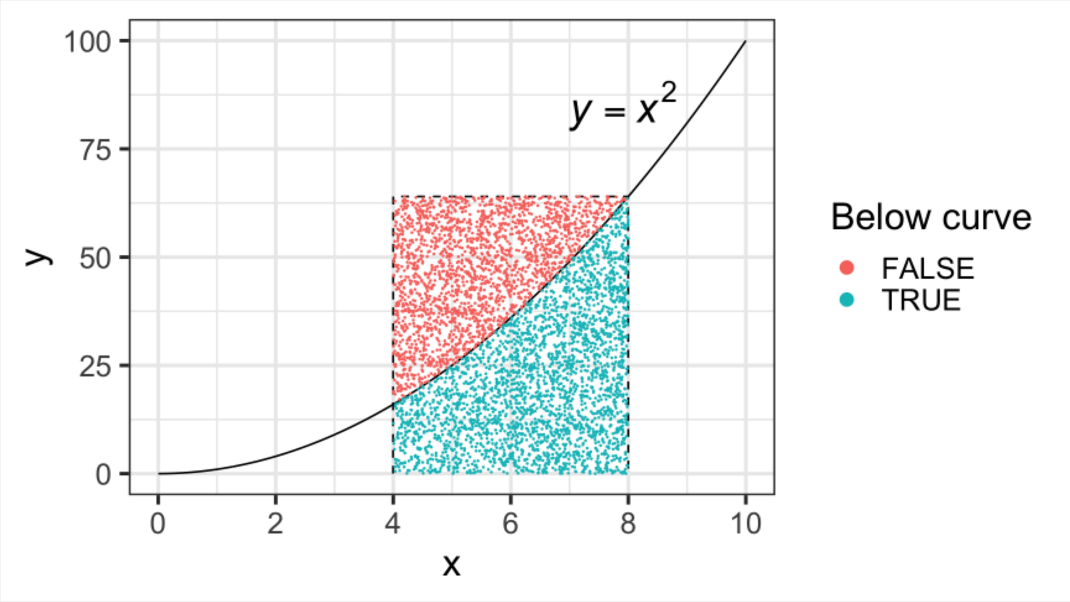
Monte Carlo Integration

Integration = compute the area "under the curve"

Find the area of $y = x^2$ between
$4 < x < 8$

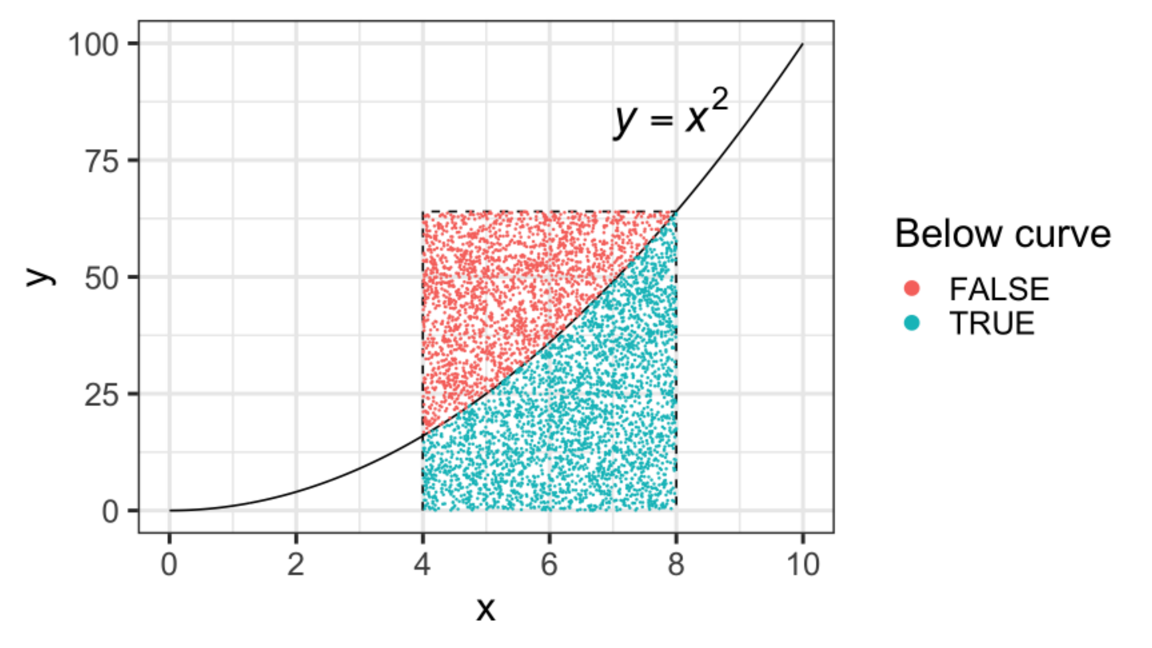


$$\frac{\text{Area Under Curve}}{\text{Area of Rectangle}} = \frac{\#\text{ Points Under Curve}}{\#\text{ Total Points}}$$



Monte Carlo Integration

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



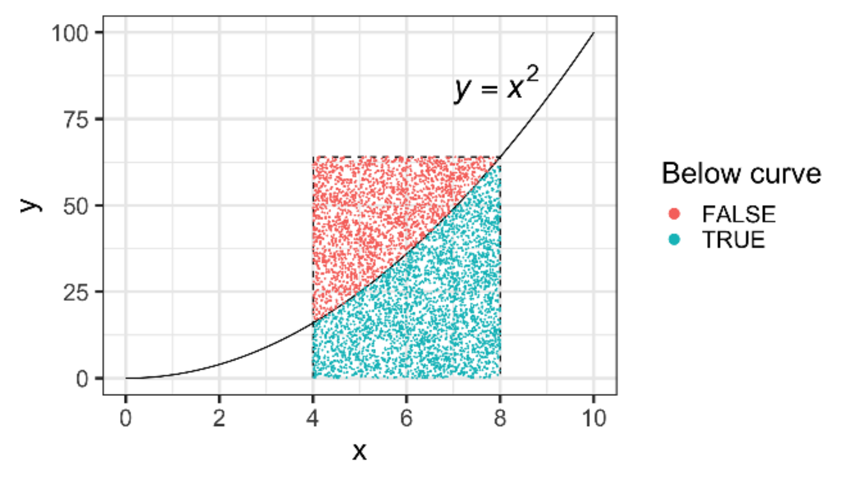
$$\text{Area Under Curve} = \text{Area of Rectangle}\left(\frac{\#\text{ Points Under Curve}}{\#\text{ Total Points}}\right)$$

Monte Carlo Integration

Step 1: Compute area of rectangle

```r
area_rectangle <- (8 - 4) * (8^2 - 0)
area_rectangle
```

```
#> [1] 256
```

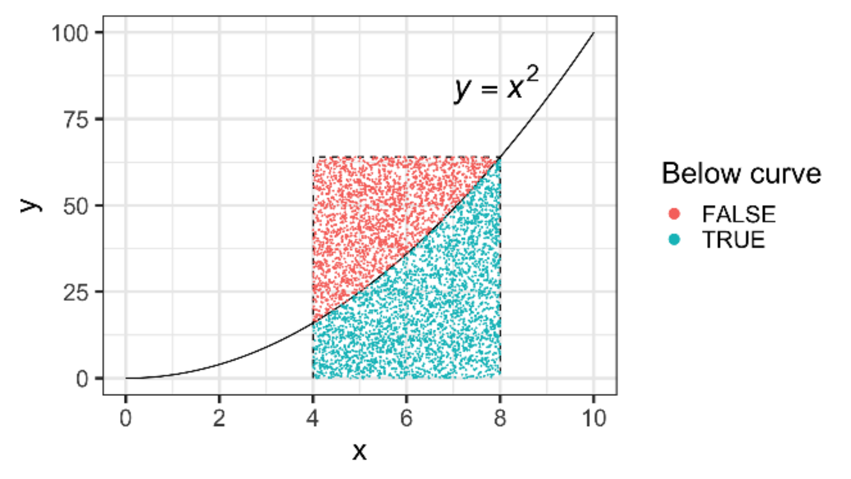


Monte Carlo Integration

Step 2: Simulate points

```
N <- 100000
points <- tibble(
    x = runif(N, min = 4, max = 8),
    y = runif(N, min = 0, max = 8^2)) %>%
    mutate(belowCurve = y < x^2)

head(points)
```

```
#> # A tibble: 6 × 3
#>        x      y belowCurve
#>    <dbl> <dbl> <lgl>
#> 1  7.59 20.6   TRUE
#> 2  4.51 25.7   FALSE
#> 3  7.46  4.37  TRUE
#> 4  4.73 56.4   FALSE
#> 5  6.43 27.1   TRUE
#> 6  7.35 48.0   TRUE
```



Monte Carlo Integration

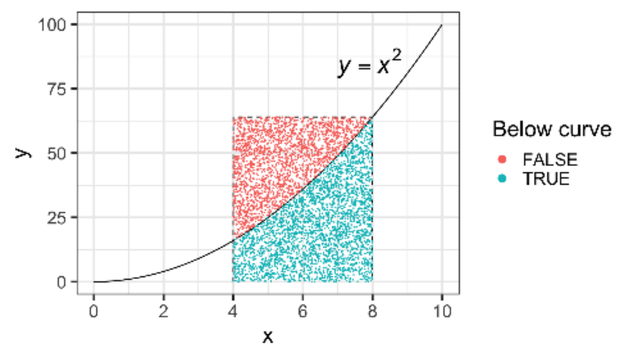
Step 3: Compute area under curve

```
N <- 100000
points <- tibble(
    x = runif(N, min = 4, max = 8),
    y = runif(N, min = 0, max = 8^2)) %>%
    mutate(belowCurve = y < x^2)

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

```
#> [1] 0.58365
```

```
area_under_curve <- area_rectangle * points_ratio
area_under_curve
```

```
#> [1] 149.4144
```



How did we do?

Simulated area under curve:

```
area_under_curve
```

```
#> [1] 149.4144
```

Actual area under curve:

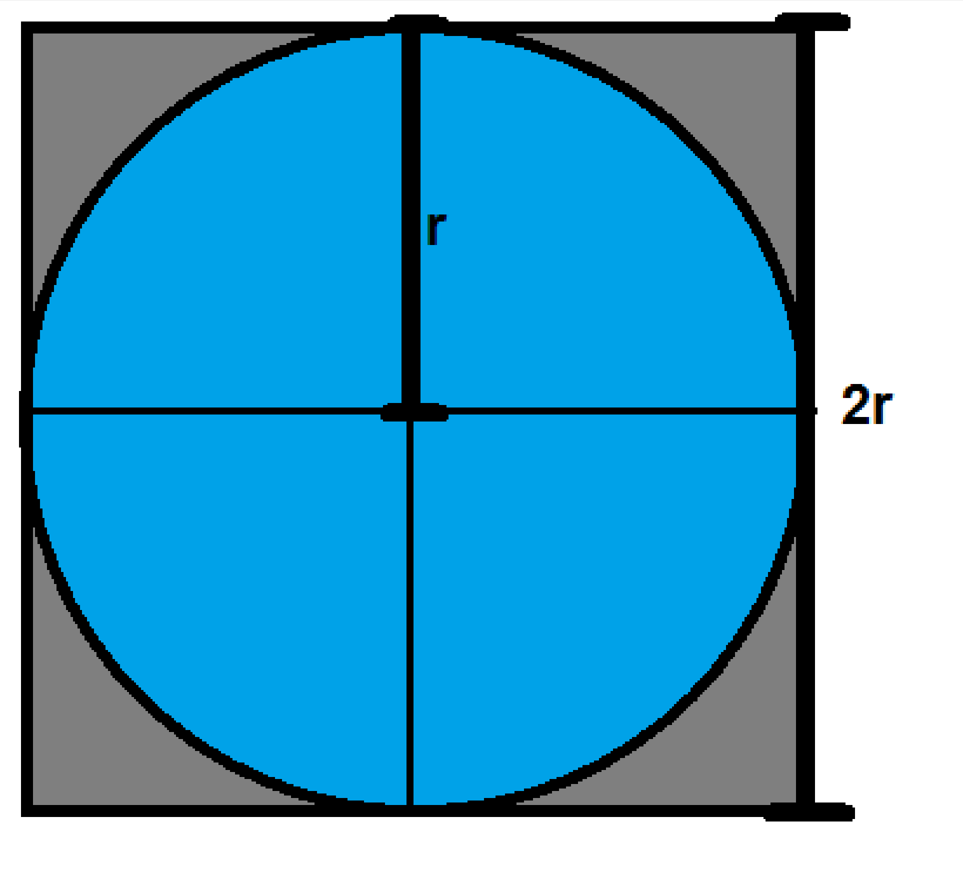
$$\int_4^8 x^2 \mathrm{dx} = \left(\frac{x^3}{3}\right)\Big|_4^8 = \frac{8^3}{3} - \frac{4^3}{3} = 149.33\bar{3}$$

% Error:

```
true_area <- ((8^3 / 3) - (4^3 / 3))
100*((area_under_curve - true_area) / true_area)
```

```
#> [1] 0.05428571
```

Monte Carlo π



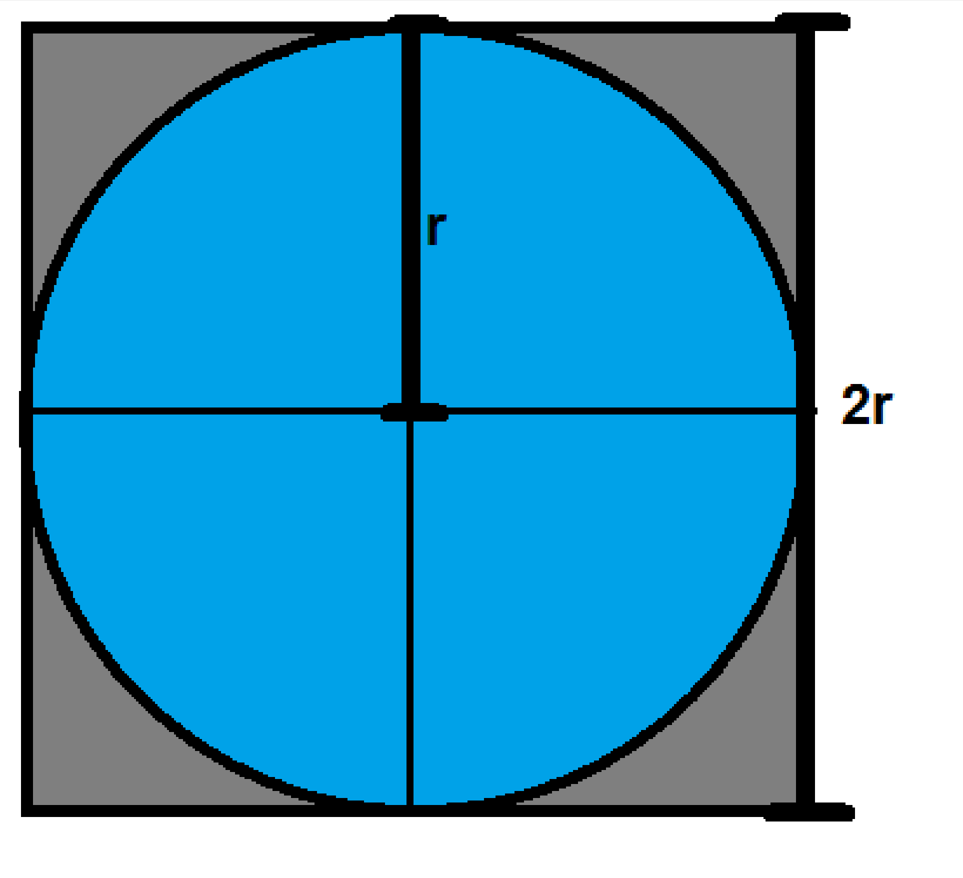
Area of a circle:

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

Area of square containing circle:

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

Monte Carlo π



Area of a circle:

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

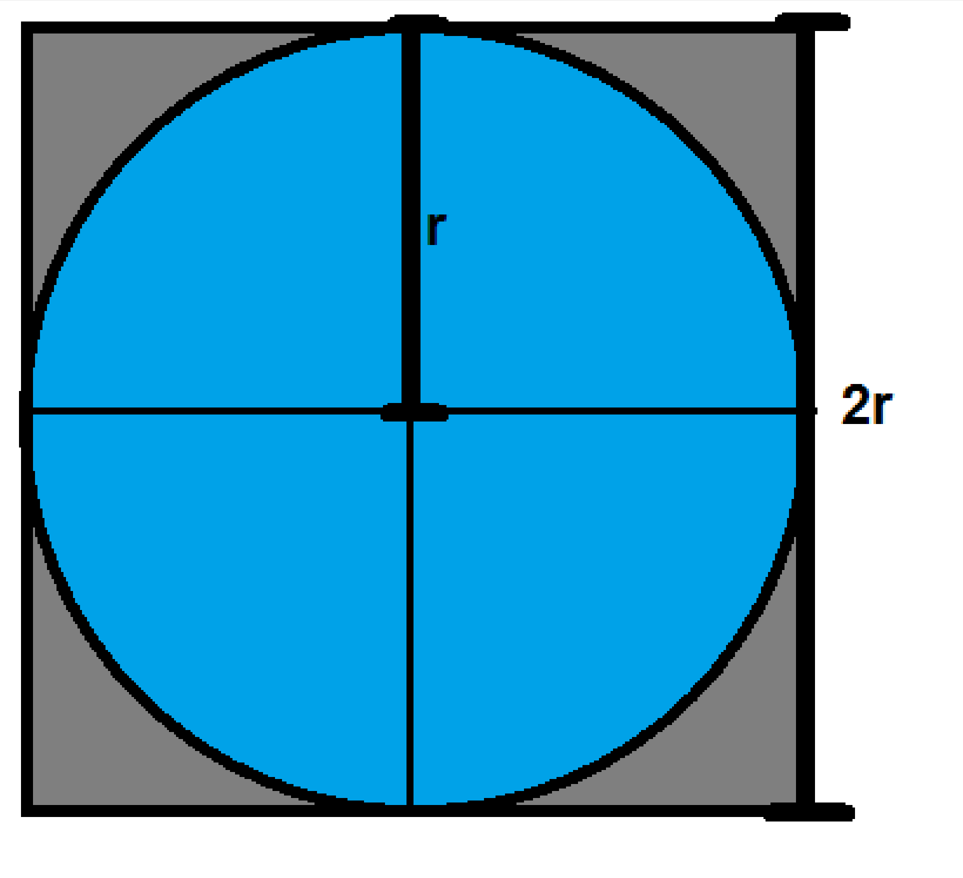
Area of square containing circle:

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

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:

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

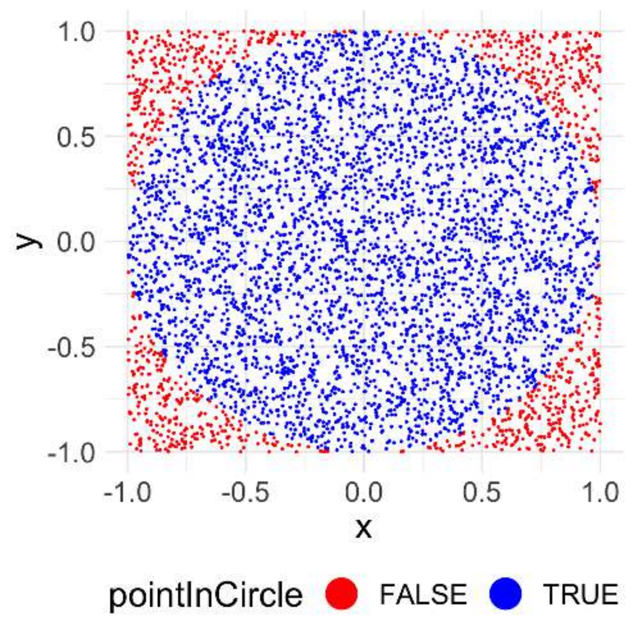
Ratio of areas = $\pi/4$:

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

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

Your Turn: Estimate π

15:00



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

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 column, `radius`, that is equal to the distance to each `(x, y)` point from the center of the square.

3. Create a new column, `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

Your Turn: Monte Hall Problem

15:00

1. You choose door 1, 2, or 3
2. One door is removed
3. Should you swap doors?

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 (`TRUE` or `FALSE`) for whether the contestant swaps doors. **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) Final is Thursday, May 11, 3:00pm-5:00pm