Simulate Pi Challenge

A short Monte Carlo simulation to estimate pi!

Some useful functions: set.seed(), runif(), paste(), sum(), mean(), sapply(), sample(), length(), hist(), abline(), function(){}, install.packages(), library()

CHALLENGE: Pi is a very useful constant. What if you didn't know the value of pi?

One way to estimate pi would be to draw a square on the ground and a perfect circle inside the square on a drizzly day. If you assume raindrops fall on random points on your drawing, you could count the number of raindrops that fall inside your square and the number of raindrops that fall inside your square AND your circle.
One easy way to decide if a raindrop is 'inside the circle' is to pick a 'center' for your circle and measure if each raindrop falls within some distance (the radius) to the center.
Remember the area of a square is length^2 and the area of a circle is pi*radius^2, and since we assume the number of raindrops is proportional to the area on the ground, we can use the ratio of raindrops that fall inside and outside of the circle to calculate pi!

Your challenge is to use R to simulate 'raindrops', or random points within a square on the x-y 2D plane. Then you'll count the points inside and outside of a certain radius and use the ratio to estimate pi. At the end you'll use a install and use a package "cowsay" to have a cow (or other animal) tell you pi. Look at the HINTS file if you want some more help along the way.
Good luck!

```
n = 100 # set n = number of points to simulate
# Test your simulation first with a small number of simulations (n),
# then when it's working increase n to get a more accurate estimate.

x = runif(n, min = -1, max = 1) # get n random x values from [-1,1]
y = runif(n, min = -1, max = 1) # get n random y values from [-1,1]
in_circle = x^2 + y^2 <= 1 # is each point in the circle with radius 1?
prop_in_circle = sum(in_circle)/length(in_circle) # calculate the proportion of points within the circl
# Area of a circle is pi*r^2
# For our circle, the area is pi because the radius = 1.
# Area of a square spanning x values [-1,1] and values [-1,1] = 2X2 = 4
# Area of our circle divided by area of our square = pi/4
pi_est = prop_in_circle * 4 # calculate pi!
pi_est
```

```
## [1] 3.16
```

Make a function that runs your simulation one time with a user-specified number of simulated raindrops/points

```
estPi <- function(i) { # my estPi() function takes in one parameter, i
  x = runif(i, min = -1, max = 1) # repeat code from above to estimate pi
  y = runif(i, min = -1, max = 1)
  in_circle = x^2 + y^2 <= 1
  prop_in_circle = sum(in_circle)/length(in_circle)
  pi_est = prop_in_circle * 4
  return(pi_est) # return estimated pi
}
```

Run your simulation multiple times, do you always get the same answer for pi?
ANSWER: No, because of the randomness in runif() I get different estimates for pi.
Find out what set.seed() does. This function is useful for reproducibility when researchers use simulations. Try setting different seeds, e.g. 5 or your favorite number, and rerunning your simulation.

```
set.seed(5) # set seed to 5, then I rerun my simulation
estPi(100)
```

```
## [1] 3.08
```

```
set.seed(37) # set seed to a new nubmer and rerun my simulation
estPi(100)
```

```
## [1] 2.84
```

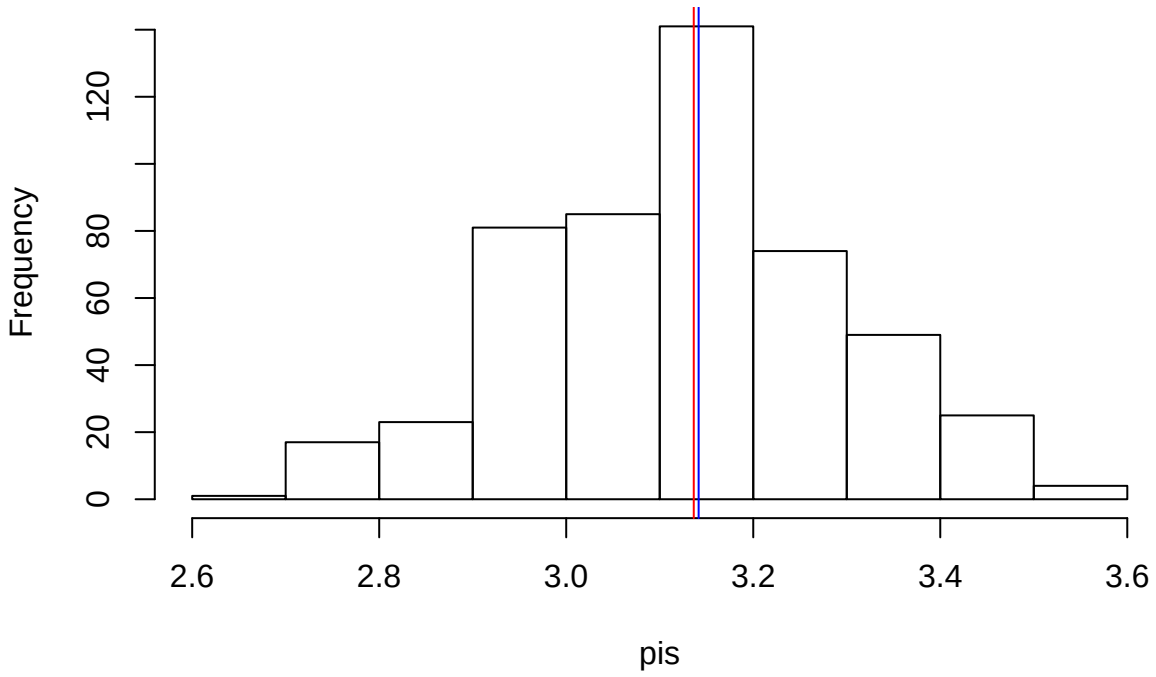
What happens if you rerun your simulation using the same seed each time?
To try this, you'll need to reset your seed inbetween each simulation run.
ANSWER: If I set differents seeds I get different answers. If I re-set to the same seed I get the same answer every time.

Run your simulation 500 times, each time simulating 100 raindrops/points

```
pis <- sapply(1:500, function(n) estPi(100)) # apply simulation 500 times
```

Plot the results using a histogram and add a vertical line for the true value of pi.
Add another vertical line (in a different color) for the mean of your simulation results.

```
hist(pis) # plot a histogram of the results
abline(v = pi, col = "blue") # add a line for 'pi' (hint: R knows many constants, including pi)
abline(v = mean(pis), col = "red") # calculate mean of simulation and add a line
```



Install the package "cowsay" and make an animal say "True pi is (value of pi)!"
The syntax to install a package is install.packages("my_package").
AFTER install, before you can use a package, remember you need to run this line of code: library(my_package).
R keeps a list of animals that you can view using names(animals) and the cowsay package lets them talk.
Not challenging enough? Have a randomly chosen animal say "I think pi is (your simulation result)". In one line of code, you should be able to generate a new simulation result and have the animal say it!

```
#install.packages("cowsay") # install package. uncomment first time you run.
library(cowsay) # load package
?cowsay # run help on cowsay package and say function
?say
names(animals) # view possible animals
```

```
##  [1] "cow"          "chicken"      "chuck"        "clippy"       "poop"
##  [6] "bigcat"       "ant"          "pumpkin"      "ghost"        "spider"
## [11] "rabbit"       "pig"          "snowman"      "frog"         "hypnotoad"
## [16] "shortcat"     "longcat"      "fish"         "signbunny"    "facecat"
## [21] "behindcat"    "stretchycat"  "anxiouscat"   "longtailcat"  "cat"
## [26] "trilobite"    "shark"        "buffalo"      "grumpycat"    "smallcat"
## [31] "yoda"         "mushroom"     "endlesshorse" "bat"          "bat2"
## [36] "turkey"       "monkey"       "daemon"       "egret"        "duckling"
## [41] "duck"         "owl"          "squirrel"     "squirrel2"
```

```
# paste sentence and pi together for an animal (e.g. 'endlesshorse') to say
say(paste0("True pi is ", pi, "!"), by = "endlesshorse")
```

```
##
##  -----
## True pi is 3.14159265358979!
##  ------
##     \
##      \
##       \
##        ,
##     _,,)\.~,,._
##      (()`  ``)\))),,_
##       |     \ ''((\))))),,_          ____
##       |6`   |   ''((\(())) "-.____.-"    `-.-,
##       |    .'\    ''))))'                  \)))
##       |   |   `.     ''                     ((((
##       \, _)     \/                          |))))
##        `'        |                          (((((
##                  \                 |       ))))))
##                   `|    |          ,\     /((((((
##                     |   / `-._____.<  \   |  )))))
##                     |   |  /         `. \  \  ((((
##                     |  / \ |           `.\  | (((
##                     \  | | |             )| |  ))
##                      | | | |             || |  '   [endless.horse]
##                      | | | |             || |
```

```
# sample one random animal name from the list
set.seed(50)
say(paste0("I think pi is ", estPi(100), "!"), by = sample(names(animals), 1))
```

```
##
##  -----
## I think pi is 2.92!
##  ------
##     \          ,'``..._   ,'``.
##      \        :,--._:)\,:,._,.:
##       \       :`--,''   :`...';\
##                `.'       `---'  `.
##               /                  :
##              /                    \
##            ,'                      :\.___,-.
##           `...,---'``````-...._    |:       \
##             (                 )   ;:    )   \  _,-.
##              `.              (   //          `'    \
##               :               `.//  )      )     , ;
##             ,-|`.            _,'/       )    ) ,' ,'
##            (  :`.`-..____..=:.-':     .     _,' ,'
##             `,'\ ``--....-)='    `._,  \  ,') _ '``._
##          _.-/ _ `.       (_)      /     )' ; / \ \`-.'
##         `--(   `-:`.     `' ___..'  _,-'   |/   `.)
##             `-. `.`.``-----``--,  .'
##               |/`.\`'        ,',); SSt
##                   `         (/  (/
##
```

Thousands of free open-source packages are posted online for use in R ... most of them significantly less silly than "cowsay" ... go explore!