In [ ]:

```r
#MONTY HALL PROBLEM:
#R(RSTUDIO) CODE ANALYSIS:

doors <-c("1","2","3")
#make a vector called doors with 3 doors A,B,C


xdata <- c()
#create an object where the data of the loop is stored into


for(i in 1:1000) {
    #everything that is to be executed from the loop goes inside the curly brackets. Into the parantheses, define how often the loop should be executed. The expression i in 1:1000 means that i will be chosen from the vector of numbers between 1 and 1000


    prize <- sample(doors,1)
    #the prize is placed into one of the doors chosen randomly. The sample() generates a random order of the three doors. The 1 takes the first element of the vector.


    pick <- sample(doors,1)
    #the pick of the participant is stored into another random generator. The sample() generated a random order of which door the participant picks. The result is saved into the object pick.


    opened_door <- sample(doors[which(doors != pick & doors != prize)],1)
    #an object called opened_door is createdin which the host of the show opens the door that was not picked by the participant and the door that is not the prize. The which command gives you a character A B or C the door that is not chosen by the contestant and the door that is not the prize. The 1 picks only one door in the case when the host could open either door.


    switchyes <- doors[which(doors != pick & doors != opened_door)]
    #simulate switching doors by taking one element that was not the initial pick and not the door that was already opened.


    if(pick==prize){xdata<-c(xdata, "noswitchwin:")}
    #these two if statements determine who wins. If the picked door contained the prize, assign the noswitchwin to the position in the xdata results vector.


    if(switchyes==prize){xdata<-c(xdata,"switchwin")}
    #if the participant switches, and it happens to be the door that contains the prize, then he won because he switched. That is why the character switchwin is assigned to that position of xdata.
}
```

```
sum(xdata=="switchwin")/length(xdata)
#computes the average of how many times the participant wins if he switc
hes the door

sum(xdata=="noswitchwin")/length(xdata)
#computes the average of how many times the participant wins if he does
n't switch doors
```

In [ ]:

```
#MONTY HALL PROBLEM:
#PYTHON(SPYDER) CODE ANALYSIS:

from numpy.random import choice     #imports the random generator functi
on

doors = ["1","2","3"]               #makes an array of the 3 doors label
ed 1,2,3
switchwin=0                         #start with 0 doors chosen
noswitchwin=0                       #start with 0 doors chosen

for _ in range(1000):               #makes a loop to make 1000 simulatio
ns
    prize=choice(doors,1)           #randomly generates a door from 1-3
 and assigns it to the prize


    pick=choice(doors,1)            #randomly generates a door from 1 to
3 and assigns it to the pick


    monty_choices= [door for door in doors if (door!=prize and door!=pic
k)]
    #if the door is not the prize and if the door is not picked


    opened_door=choice(monty_choices,1)
    #open one door randomly with the condition above


    switched_door= [door for door in doors if (door!=opened_door and doo
r!=pick)]
    #if the door is not opened and if the door is not picked, the partic
iapnt has a choice to switch doors

    if pick == prize:               #if the pick was the prize
        noswitchwin += 1            #no need to switch to win
    else:
        switchwin += 1              #but if the pick was not the prize,
 you will win if you switch

switchwin/1000
#calculates the average of the times you will win if you switch

noswitchwin/1000
#calculates the average of the times you will win if you do not switch
```

In [ ]:

```
#BIRTHDAY PARADOX:
#MONTE CARLO SIMULATION CODE (WORKED OUT IN RSTUDIO NOT JUPYTER)
#THE SCREENSHOT OF THE CODE WORKING IN RSTUDIO IS ON LAST PAGE

class_size <- 35;
simulations <- 10000;
results <- numeric(simulations)

for (i in 1:simulations)  {
  b <- sample(1:365, class_size, repl=T)
  results[i] <- class_size - length(unique(b))  }

mean(results)       #gives the average of the results in the array from the loop
mean(results==0)    #gives the probability of the results in the array from the loop
```

In [7]:

```
#BIRTHDAY PARADOX:
#ALGORITHM TO SEE AROUND WHICH CLASS SIZE IS REQUIRED TO HAVE A 50-50 CHANCE OF SHARING BIRTHDAYS
#THIS CODE IS FOR JUPYTER NOT RSTUDIO

import math as math
import sys

fact = math.factorial(365)
result = 0
desired_probability = 50

for i in range (1,366):
    a = 365**i
    b = math.factorial(365-i)
    result = (1-(fact / (a*b)))*100
    if (result > desired_probability):
        print ('In order for a 50/50 chance of sharing birthdays, the class size required is ',i)
        sys.exit()
```

```
In order for a 50/50 chance of sharing birthdays, the class size required is  23

An exception has occurred, use %tb to see the full traceback.

SystemExit
```

Monty Hall Problem

Pencil/Pen Approach

1) probability of any of these three events:

$P(a) = \frac{1}{3}$

$P(b) = \frac{1}{3}$

$P(c) = \frac{1}{3}$

CAR	goat	goat
1	2	3

$$P(C|a) = \frac{P(C \cap a)}{P(a)} = \frac{(\frac{1}{2})(\frac{1}{3})}{(\frac{1}{3})} = \boxed{\frac{1}{2}}$$ 50%

2)

goat	CAR	goat
1	2	3

$$P(C|b) = \frac{P(C \cap b)}{P(b)} = \frac{(\frac{2}{2})(\frac{1}{3})}{(\frac{1}{3})} = \boxed{1}$$ 100%

3)

goat	goat	CAR
1	2	3

$$P(C|c) = \frac{P(C \cap c)}{P(c)} = \frac{(0)(\frac{1}{3})}{(\frac{1}{3})} = \boxed{0}$$ 0%

4) $P(C) =$ 50%, 100% or 0%
depends on picked door and prize

Baye's Theorem

1). $$P(a|C) = \frac{P(C|a)\,P(a)}{P(C|a)\,P(a) + P(C|b)\,P(b) + P(C|c)\,P(c)}$$

$$= \frac{(\frac{1}{2})(\frac{1}{3})}{(\frac{1}{2})(\frac{1}{3}) + (1)(\frac{1}{3}) + (0)(\frac{1}{3})}$$

$$= \boxed{\frac{1}{3}}$$

2) $$P(b|C) = \frac{P(C|b)\,P(b)}{P(C|a)\,P(a) + P(C|b)\,P(b) + P(C|c)\,P(c)}$$

$$= \frac{(1)(\frac{1}{3})}{(\frac{1}{2})(\frac{1}{3}) + (1)(\frac{1}{3}) + (0)(\frac{1}{3})}$$

$$= \boxed{\frac{2}{3}}$$

This means the chance of winning the prize if you switch is higher than if you were to not switch doors.

Birthday Paradox
Pencil / Pen Approach

1). $\frac{365}{365} = \boxed{1}$ OR 100% this is because this person is the only one in the room.

$P(A) = 1$

2). $P(B \mid A) = \frac{P(B \cap A)}{P(A)} = \frac{\left(\frac{364}{365}\right)(1)}{1} = \boxed{\frac{364}{365}} \approx 0.9973$

3) $P(C \mid (A \cap B)) = \frac{P(C \cap (A \cap B))}{P(A \cap B)} = \frac{\left(\frac{363}{365}\right)\left(\frac{364}{365}\right)\left(\frac{365}{365}\right)}{\frac{364}{365}}$

$= \boxed{\frac{363}{365}} \approx 0.9945$

4) $P(D \mid (A \cap B \cap C)) = \frac{P(D \cap P(A \cap B \cap C))}{P(A \cap B \cap C)}$

$= \frac{\left(\frac{362}{365}\right)\left(\frac{363}{365}\right)\left(\frac{364}{365}\right)\left(\frac{365}{365}\right)}{\left(\frac{363}{365}\right)\left(\frac{364}{365}\right)\left(\frac{365}{365}\right)}$

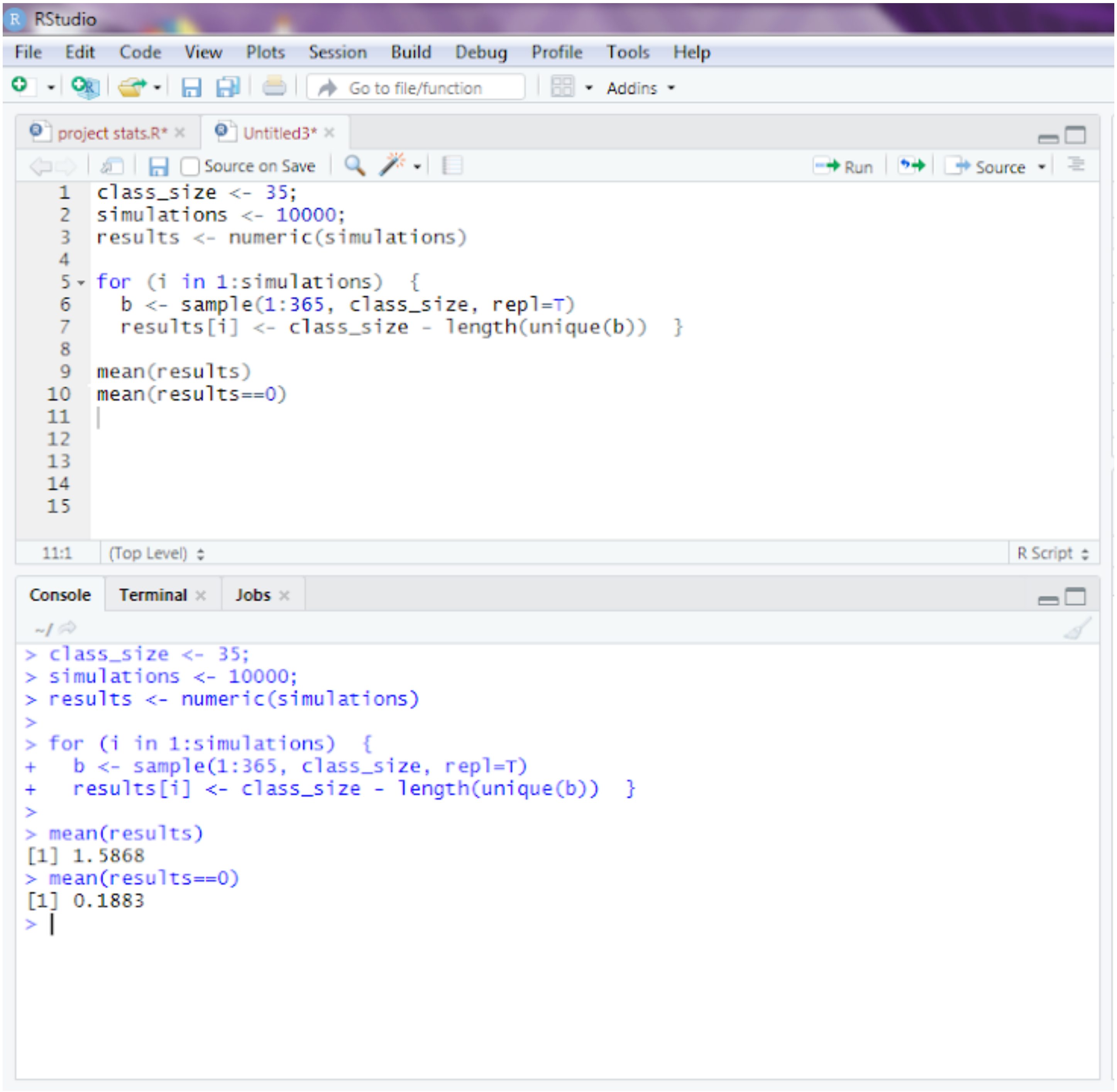
$= \boxed{\frac{362}{365}} \approx 0.9918$

5) $\frac{365 - n + 1}{365}$ gives you the probability of the nth person to enter having a unique birthday

6). $\frac{365 - 35 + 1}{365} \approx 0.9068$ or $\boxed{90.68\%}$ having a unique birthday

7) $1 - P(B \mid A) = 1 - 0.99726 \approx .002739$

$\left(\frac{365}{365}\right)\left(\frac{1}{365}\right) \approx 0.0027$ OR $\boxed{0.2739\%}$



```r
class_size <- 35;
simulations <- 10000;
results <- numeric(simulations)

for (i in 1:simulations)  {
  b <- sample(1:365, class_size, repl=T)
  results[i] <- class_size - length(unique(b))  }

mean(results)
mean(results==0)
```

```
> class_size <- 35;
> simulations <- 10000;
> results <- numeric(simulations)
>
> for (i in 1:simulations)  {
+   b <- sample(1:365, class_size, repl=T)
+   results[i] <- class_size - length(unique(b))  }
>
> mean(results)
[1] 1.5868
> mean(results==0)
[1] 0.1883
>
```