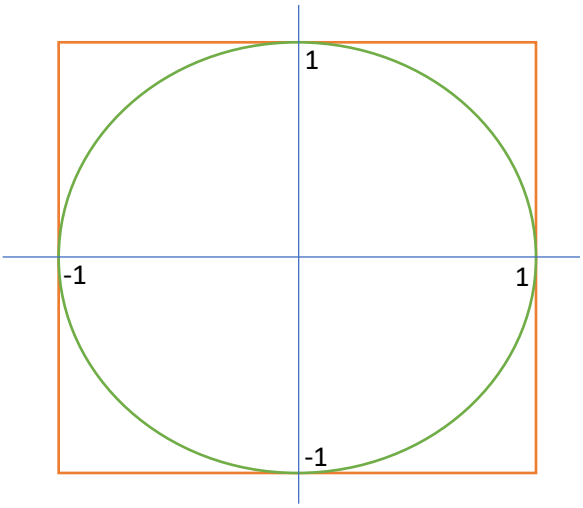
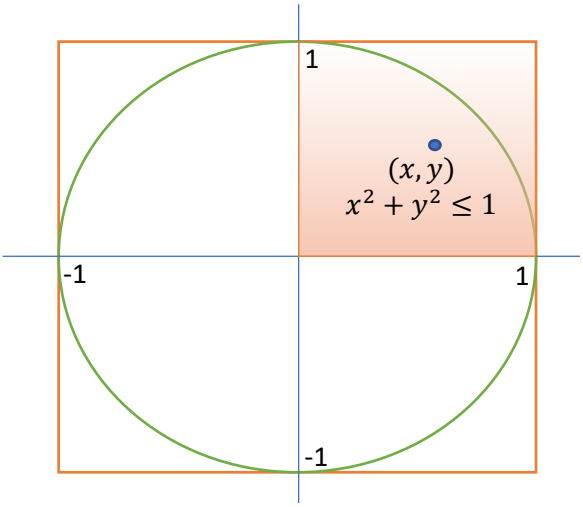
Monte Carlo Simulation: Estimating Pi

In this handout, we will consider a method for estimating the value of π using a Monte Carlo simulation in both Excel and in Python.

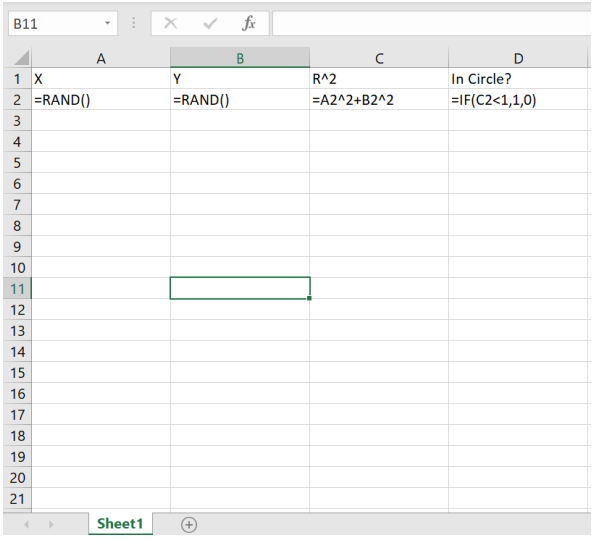
First, Excel. We'll use this section to set up the mathematical ideas. If you want to know how the set-up in Python works, you'll still need to read this section.



Consider the unit circle. It has a radius of 1, and so using the area formula, $A = \pi r^2$, the area is π. In Excel, and many calculator programs, the default random uniform number is from 0 to 1, we are going to only consider the first quadrant of the circle where x and y are both positive. Any point in that unit square in the first quadrant can be generated by some number between 0 and 1 for the first component, and a second such number for the second. Any pair of points inside the circle will satisfy the inequality $x^2 + y^2 \leq 1$, while any points outside the circle will not. The area inside the circle is $\frac{1}{4}$ of the total area of the circle, and so is equal to $\frac{\pi}{4} \approx 0.785398\ldots$. compared to the area of the whole square, which is 1. We will count the number of randomly generated points inside the circle and find the ratio to the total number of points generated: $\frac{number\ of\ points\ inside\ the\ circle}{total\ number\ of\ pairs}$. That proportion will be our estimate for $\frac{\pi}{4}$, and then we'll multiply by 4 to estimate π.



Note: You could also find the number that falls outside the circle (it would be a smaller count), but then you'd have to subtract from 1 to get the proportion inside the circle. However, we'll stick with the direct calculation.



To set up Excel for this simulation, first, we need the formulas for the random points, our x and y coordinates. In Excel, we use RAND() to generate the random numbers between 0 and 1. We need one each for x and y. The third column is then the calculation for the distance from the origin: the $x^2 + y^2$ value. And then we check to see if it's inside the circle. In practice, the chance that you will

Betsy McCall

get a point that falls *exactly* on the circle is basically zero, so I've just used $<$ instead of $\leq$. It's easy to flip the inequality here, but then also change the results of your condition to =IF(C2>1,0,1) or you'll be counting outside the circle instead of inside. This last formula evaluates whether the result of the pair is inside or outside the circle (1 for is in inside and 0 for is not inside).

When you are finished, copy the formula down until you have enough for an initial simulation. The first time through, don't make it too big until you are sure it works, then make it larger (longer). My final final has 20,000 entries. Excel can handle up to a million lines, but be cautious about this because depending on the speed of your computer, you could crash Excel (I've done that, too), and every time you do anything, it will recalculate all the random numbers, so it's best to start small, and then when everything is set up, then go big.

F14

	A	B	C	D	E
1	X	Y	R^2	In Circle?	
2	=RAND()	=RAND()	=A2^2+B2^2	=IF(C2<1,1,0)	
3	=RAND()	=RAND()	=A3^2+B3^2	=IF(C3<1,1,0)	
4	=RAND()	=RAND()	=A4^2+B4^2	=IF(C4<1,1,0)	
5	=RAND()	=RAND()	=A5^2+B5^2	=IF(C5<1,1,0)	
6	=RAND()	=RAND()	=A6^2+B6^2	=IF(C6<1,1,0)	
7	=RAND()	=RAND()	=A7^2+B7^2	=IF(C7<1,1,0)	
8	=RAND()	=RAND()	=A8^2+B8^2	=IF(C8<1,1,0)	
9	=RAND()	=RAND()	=A9^2+B9^2	=IF(C9<1,1,0)	
10	=RAND()	=RAND()	=A10^2+B10^2	=IF(C10<1,1,0)	
11	=RAND()	=RAND()	=A11^2+B11^2	=IF(C11<1,1,0)	
12	=RAND()	=RAND()	=A12^2+B12^2	=IF(C12<1,1,0)	
13	=RAND()	=RAND()	=A13^2+B13^2	=IF(C13<1,1,0)	
14	=RAND()	=RAND()	=A14^2+B14^2	=IF(C14<1,1,0)	
15	=RAND()	=RAND()	=A15^2+B15^2	=IF(C15<1,1,0)	
16	=RAND()	=RAND()	=A16^2+B16^2	=IF(C16<1,1,0)	
17	=RAND()	=RAND()	=A17^2+B17^2	=IF(C17<1,1,0)	
18	=RAND()	=RAND()	=A18^2+B18^2	=IF(C18<1,1,0)	
19	=RAND()	=RAND()	=A19^2+B19^2	=IF(C19<1,1,0)	
20	=RAND()	=RAND()	=A20^2+B20^2	=IF(C20<1,1,0)	
21	=RAND()	=RAND()	=A21^2+B21^2	=IF(C21<1,1,0)	

Sheet1

Once several points are set up, we want to summarize the results to get our estimate for π. We need to count the number of 1's in the D column, divide by the total number of runs (the number of pairs attempted), and then convert that to our estimate. I've also calculated the percent error for the run.

The new formulas look like this:

	E	F
	Sum	=SUM(D:D)
	Proportion	=F2/COUNT(D:D)
	Estimate for Pi	=F3*4
	Error %	=ABS((PI()-F4)/PI())

I've set up the sum and the denominator in the proportion to consider the whole column, so that when the simulation gets extended to a larger number of attempts, the denominator will get updated automatically.

	A	B	C	D	E	F
1	X	Y	R^2	In Circle?		
2	0.435641	0.396544	0.34703	1	Sum	18
3	0.738572	0.401868	0.706987	1	Proportion	0.72
4	0.921185	0.169537	0.877325	1	Estimate for Pi	2.8800000000
5	0.995943	0.668829	1.439234	0		
6	0.051261	0.742807	0.55439	1	Error %	8.3267528%
7	0.480461	0.94767	1.128922	0		
8	0.908264	0.957822	1.742368	0		

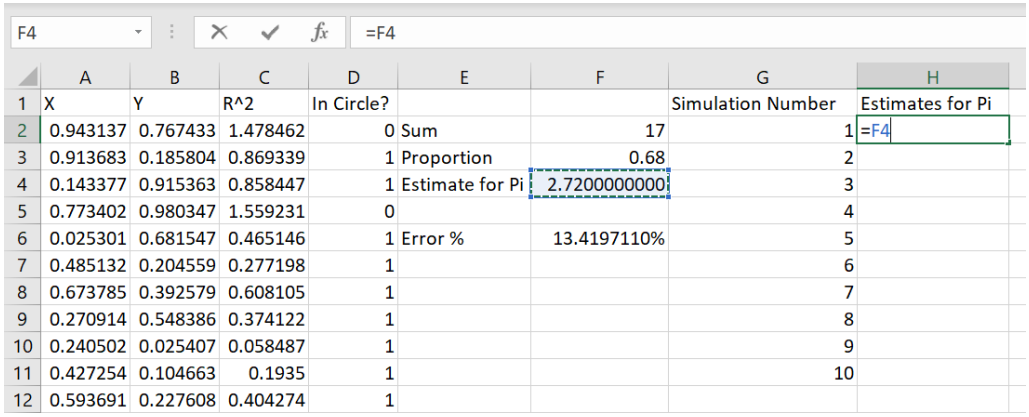
For a small run of just 25 points, the estimates vary wildly. One such run gave me, which isn't great. For a run of 20,000, though, I got something a bit better.

This latter one round to 3.14 so we've got about 2 decimal places accuracy. We need more runs if we want a better estimate.

	A	B	C	D	E	F
1	X	Y	R^2	In Circle?		
2	0.274587	0.172107	0.105019	1	Sum	15688
3	0.083324	0.427698	0.189869	1	Proportion	0.7844
4	0.530585	0.302239	0.372869	1	Estimate for Pi	3.1376000000
5	0.933768	0.592277	1.222714	0		
6	0.601233	0.198855	0.401025	1	Error %	0.1270901%
7	0.926193	0.104688	0.868792	1		

One way to do this is to extend the table, but there is a more compact way to do this. We can use the Data Table to record the results of various simulations and then since they are all the same size, average them together to obtain a result equivalent to a larger simulation. Excel will actually calculate the simulations, but they will take up less space in the sheet than just copying out the rows.

To set up your data table, first I created a column with the number of simulation runs (this is optional, but useful later when you want to count them easily). And a second column beside it with the outcome of the simulation. Write a formula in the top entry pointing at the estimate for π in your previous calculation.

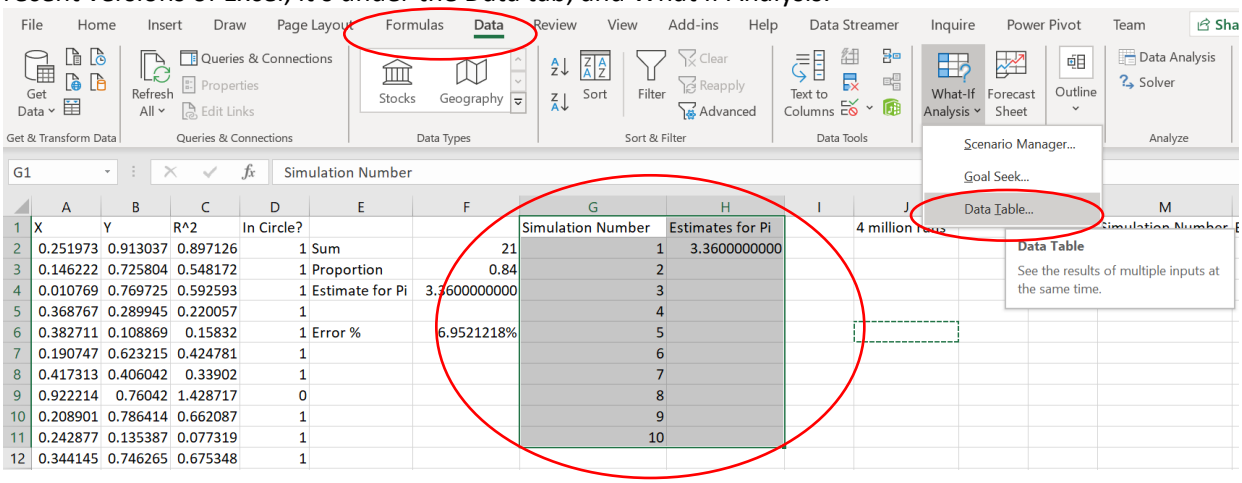


F4 =F4

	A	B	C	D	E	F	G	H
1	X	Y	R^2	In Circle?			Simulation Number	Estimates for Pi
2	0.943137	0.767433	1.478462	0	Sum	17	1	=F4
3	0.913683	0.185804	0.869339	1	Proportion	0.68	2	
4	0.143377	0.915363	0.858447	1	Estimate for Pi	2.7200000000	3	
5	0.773402	0.980347	1.559231	0			4	
6	0.025301	0.681547	0.465146	1	Error %	13.4197110%	5	
7	0.485132	0.204559	0.277198	1			6	
8	0.673785	0.392579	0.608105	1			7	
9	0.270914	0.548386	0.374122	1			8	
10	0.240502	0.025407	0.058487	1			9	
11	0.427254	0.104663	0.1935	1			10	
12	0.593691	0.227608	0.404274	1				

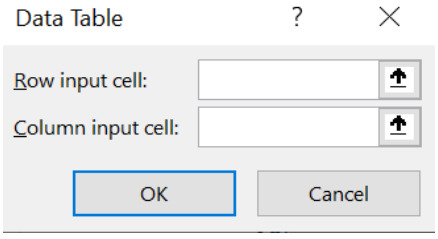
We'll just do 10 simulations for now to keep the spreadsheet updating relatively quickly.

First, highlight the column starting at G2 and down to H11. Then, find the Data Table option. In more recent versions of Excel, it's under the Data tab, and What If Analysis.



	A	B	C	D	E	F	G	H
1	X	Y	R^2	In Circle?			Simulation Number	Estimates for Pi
2	0.251973	0.913037	0.897126	1	Sum	21	1	3.3600000000
3	0.146222	0.725804	0.548172	1	Proportion	0.84	2	
4	0.010769	0.769725	0.592593	1	Estimate for Pi	3.3600000000	3	
5	0.368767	0.289945	0.220057	1			4	
6	0.382711	0.108869	0.15832	1	Error %	6.9521218%	5	
7	0.190747	0.623215	0.424781	1			6	
8	0.417313	0.406042	0.33902	1			7	
9	0.922214	0.76042	1.428717	0			8	
10	0.208901	0.786414	0.662087	1			9	
11	0.242877	0.135387	0.077319	1			10	
12	0.344145	0.746265	0.675348	1				

The following dialog box will pop up. We aren't really adding any inputs, so you can leave the Row input cell blank, and the Column input cell, just click on a blank cell off to the side in the I or J columns.



Excel will then recalculate all the formulas, here 10 times, one for each position in the table, and record the results.

	A	B	C	D	E	F	G	H
1	X	Y	R^2	In Circle?			Simulation Number	Estimates for Pi
2	0.869169	0.360502	0.885417	1	Sum	20	1	3.2000000000
3	0.936898	0.289754	0.961735	1	Proportion	0.8	2	3.52
4	0.580383	0.816754	1.003931	0	Estimate for Pi	3.2000000000	3	3.2
5	0.660317	0.379403	0.579965	1			4	3.68
6	0.939508	0.117913	0.896578	1	Error %	1.8591636%	5	3.52
7	0.943236	0.376265	1.03127	0			6	3.36
8	0.2599	0.184992	0.10177	1			7	2.56
9	0.649766	0.447221	0.622202	1			8	3.2
10	0.599467	0.192681	0.396486	1			9	3.36
11	0.545888	0.032589	0.299056	1			10	3.52
12	0.62093	0.11749	0.399358	1				

If you then average these results, you can obtain an estimate of π equivalent to $25 \times 10 = 250$ simulations. In my attached Excel sheet, my initial simulation was $20{,}000$ pairs, and I did 200 times that many runs in my table producing the equivalent of $20{,}000 \times 200 = 4{,}000{,}000$ pairs of points. You can then do additional Data Tables based on the results of your data table and continue multiplying. I did a second data table from the result of the first, to obtain an estimate from $20{,}000 \times 200 \times 200 = 800{,}000{,}000$ pairs to obtain three decimal places accuracy.

When you open the Excel file, give it time to calculate all 1.6 billion random numbers. Depending on the speed of your computer, that may take a little while.

In Python. Python is a little different than Excel, but the math is fundamentally the same. I built my simulation in Jupyter Notebook. I'm using the Anaconda Release of Python 3. All the libraries used are standard. Start by importing some needed libraries.

```
import random
import math
```

Then, we'll need to write a little program to do the calculations. I've written this one so that we can adjust the number of simulated pairs we are calculating.

```
def estPi(sims):
    count=0
    for i in range(sims):
        x=random.uniform(0.0,1.0)
        y=random.uniform(0.0,1.0)
        r2=x*x+y*y
        if r2<1:
            count+=1
    pi_est=(count/sims)*4
    return pi_est
```

Let's break down the steps here a little bit. First, we define the function and its inputs. Sims will be the number of simulations to run, and the name of the function is estPi. The next line sets the count variable to be zero before getting started.

Next, we use a loop to run through the required number of simulations. Each time through the loop two uniform random numbers are generated in the range $(0,1)$, just like in Excel, and the radius (the distance from the origin) is calculated. The next if statement checks, like Column D did in Excel, to see if the distance calculation puts it inside the circle. If so, the count adds 1, if not nothing happens. Then the loop repeats the simulation.

Finally, the count is divided by the total number of simulations and multiplied by 4 to obtain an estimate for π.

```
estPi(100)
```

```
3.12
```

To run the simulation, run the function with the number of simulations to do, here it's 100. The output provides the estimate.

If you run a larger simulation, like $100{,}000{,}000$ times, expect to wait a bit for the calculation to go through. If you set the output equal to a variable, you can then calculate the error of your estimate.

```
p=estPi(100000000)
```

```
error=abs((math.pi-p)/math.pi)
(p, error)
```

```
(3.14175104, 5.041596020599311e-05)
```

My Excel sheet converted errors to percentages, but here, I just left them as decimals.

One note about Python is that you can't use scientific notation for simulation number unless you adjust a program a little. Python will interpret something like $10e8$ as a float, and not an integer, and the loop requires an integer.

You might be wondering why we might want to estimate π like this. The truth is that there are many better ways to estimate π. But this one is easy to understand, whereas some of the others require significantly more mathematics to know what is going on, but they do yield results with less computational intensity.