ENGG1003 - Thursday Week 4

Using random numbers, and reading from spreadsheets

Steve Weller & Sarah Johnson

University of Newcastle

18 March, 2021

Last compiled: March 18, 2021 2:19pm +11:00

Lecture overview

1. Using random numbers
2. Reading from spreadsheets

- More practice with arrays, iteration, conditional code execution (`if`) and plotting

1) Using random numbers

Recap: generating random numbers

```
In [1]: import numpy as np

In [2]: np.random.randint(1, 6, 4)      # ...4 integers from [1, 6)
Out[2]: array([1, 3, 5, 3])

In [3]: np.random.random(4)          # ...4 floats from [0, 1)
Out[3]: array([ 0.79183276,  0.01398365,  0.04982849,  0.11630963])

In [4]: np.random.uniform(10, 20, 4)    # ...4 floats from [10, 20)
Out[4]: array([ 10.95846078,  17.3971301 ,  19.73964488,  18.14332234])
```

Using `numpy` library:

- random integers
- random floats from $[0, 1)$
- random floats from $[a, b]$

Random integers: simulating coin toss

Example 1

Simulate the toss of a coin N times as follows:

- generate a length-N array of randomly chosen 0s and 1s
  - $0 =$ heads, $1 =$ tails
  - equally likely heads and tails ie: fair coin
- display *expected* number of heads observed
- display *actual* number of heads observed
- test/debug with $N = 100$, then $N = 100,000$

Coin toss simulation

```
1  import numpy as np
2
3  # generate random array of 0s and 1s
4  # 0==heads & 1==tails
5  # N integers from [0,2) ie: 0 or 1
6  N = 100000
7  x = np.random.randint(0, 2, N)
8  print(x)
9
10 headCnt = 0;
11 for i in range(0,N,1):
12     if x[i]==0:
13         headCnt += 1
14
15 print('Expected number of heads: {}'.format(N/2))
16 print('Observed number of heads: {}'.format(headCnt))
```

- Live demo of `headsTails.py`

Random floats: engineering tolerance

Example 2

- steel bolts manufactured with length uniformly distributed between 17 mm and 19 mm
- only bolts with length in range $[17.25, 18.75]$ mm are within tolerance ie: acceptable
- write Python code with random numbers to estimate percentage of bolts which are acceptable
  - demonstrate your code for $N = 10,000$
  - compare with expected percentage of acceptable bolts:

$$100 \times \frac{18.75 - 17.25}{19 - 17} = 75\%$$

Engineering tolerance simulation

```
1  import numpy as np
2
3  # generate random array of N floats in range [17,19]
4  N = 10000
5  x = np.random.uniform(17,19,N)
6  tolLow = 17.25
7  tolHigh = 18.75
8  #print(x)
9
10 goodCnt = 0;
11 for i in range(0,N,1):
12     if tolLow <= x[i] <= tolHigh:
13         goodCnt += 1
14
15 print('Percentage of parts within tolerance: {}%'.
       format(100*goodCnt/N))
```

- Live demo of `engTolerance.py`

Random floats: simulate dartboard

Example 3

- a circular dartboard radius 1m is centred in the middle of a square, side length 2 m
- darts are thrown randomly and land with uniform probability in the square
  - some (but not all) darts land inside the circle
- write a Python program which puts a red dot if the dart lands inside (or on the perimeter) of the circle, and a blue dot otherwise
  - Hint: points inside (or on perimeter) of circle satisfy

$$x^2 + y^2 \leq 1$$

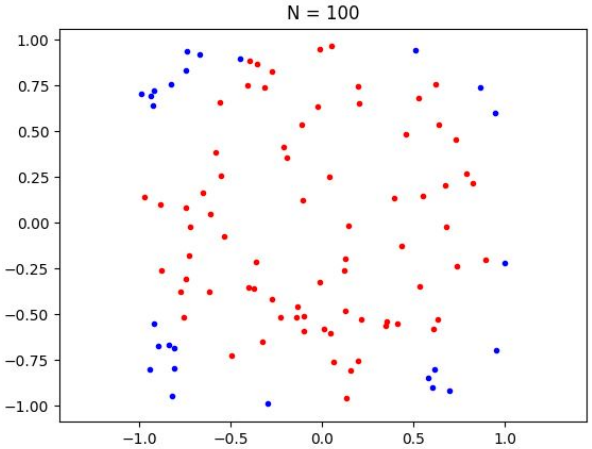
- run your program with $N = 100$, $N = 1000$ and $N = 10,000$

Dartboard simulation: code

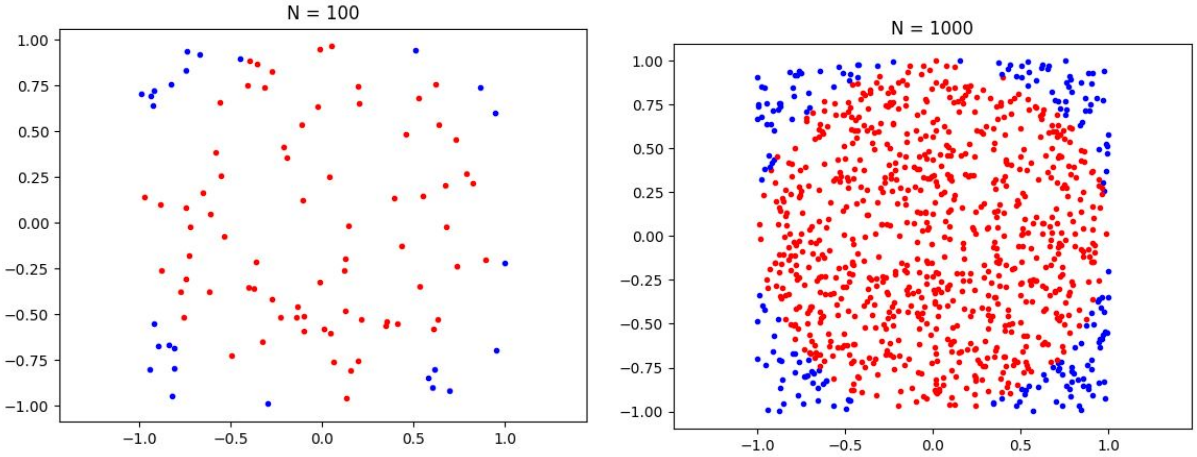
```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  # generate random array of (x,y) pairs covering
5  # square with edge length 2
6  N = 10000
7  x = np.random.uniform(-1, 1, N)   # N floats from [-1,1]
8  y = np.random.uniform(-1, 1, N)   # N floats from [-1,1]
9
10 insideCnt = 0;
11 for i in range(0,N,1):
12     if x[i]**2 + y[i]**2 <= 1:
13         plt.plot(x[i],y[i],'r.')
14     else:
15         plt.plot(x[i],y[i],'b.')
16
17 plt.axis('equal')   # plot with aspect ratio 1:1
18 plt.show()
```

- Live demo of `dartboard.py`

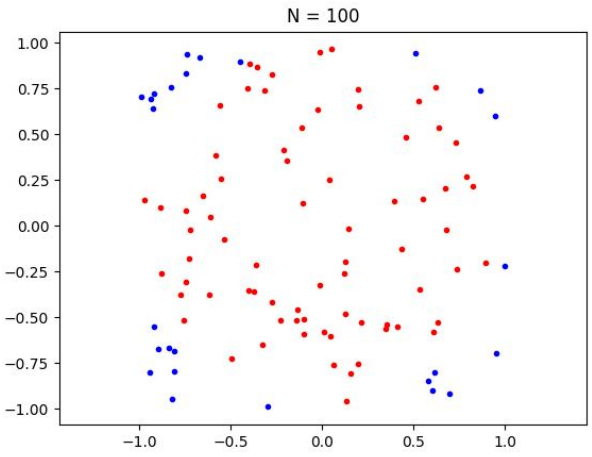
Dartboard simulation: output

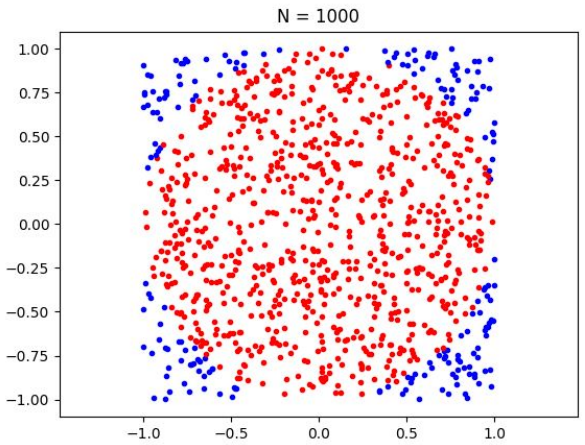


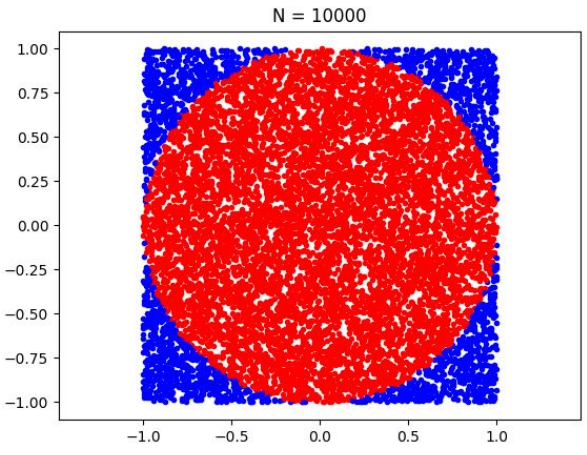
Dartboard simulation: output



Dartboard simulation: output







Random floats: estimate π

Example 4

- modify previous example to count points inside circle, hence...
- estimate π

Random floats: estimate π

```python
import numpy as np
import matplotlib.pyplot as plt

# generate random array of (x,y) pairs covering
# square with edge length 2
N = 10000
x = np.random.uniform(-1, 1, N)      # N floats from [-1,1]
y = np.random.uniform(-1, 1, N)      # N floats from [-1,1]

insideCnt = 0;
for i in range(0,N,1):
    if x[i]**2 + y[i]**2 <= 1:
        plt.plot(x[i],y[i],'r.')
        insideCnt += 1
    else:
        plt.plot(x[i],y[i],'b.')

R = insideCnt/N
print('Estimate of pi: {}'.format(4*R))

plt.axis('equal')
plt.show()
```

Live demo

2) Reading from spreadsheets

- so far have been working "toy problems" with small data sets
- engineers and scientists often work with *large* datasets
  - spreadsheets eg: CSV or XML files
  - databases eg: SQL files
  - wide range of other software packages
- there's a package for that: it's called `pandas`

  ...and we can import it just like `numpy` and `matplotlib`

```
1 import pandas as pd
```

Working with Data

- To import the data is then relatively straightforward
- For a CSV file the instruction is

```
1 import pandas as pd
2 mydata = pd.read_csv('filename.csv')
```

- If the CSV file is not in the same folder as the python project you can specify its location with a path

```
1 import pandas as pd
2 mydata = pd.read_csv('c:\Myfolder\filename.csv')
```

Working with Data

- Pandas will import the contents of the spreadsheet into something called a data structure (which we won't spend any time on today but may get back to later in the course)
- To check on the data imported into 'mydata' the following instruction is very useful

```
1 print(mydata.head())
```

- This prints out the first few lines of the data set

Working with Data

- One final very useful instruction will extract a column of the data and save it as a numpy array using the column name in the original spreadsheet

```
1 mycolumn = mydata['column_name'].values
```

- We can now process this data using the python skills we have been building in this course

Example

- We have a csv spreadsheet (Rainfall.csv) which records annual rainfall for 20 regions over the last 10 years

	A	B	C	D	E	F	G
1	Region	Rainfall 2010	Rainfall 2012	Rainfall 2014	Rainfall 2016	Rainfall 2018	Rainfall 2020
2	A	67	78	70	78	63	65
3	B	74	79	79	79	64	74
4	C	75	80	80	80	68	60
5	D	76	81	81	81	69	61
6	E	77	67	67	67	67	62
7	F	78	68	68	68	90	63
8	G	79	79	79	79	64	64
9	H	80	80	80	80	68	68
10	I	81	67	67	67	65	69
11	J	67	74	74	78	74	67
12	K	68	75	75	79	60	90
13	L	60	76	76	80	61	60
14	M	68	77	75	81	62	68
15	N	69	69	69	67	60	60
16	O	70	70	63	68	34	34
17	P	65	65	77	79	65	65
18	Q	66	66	69	80	57	57
19	R	67	67	70	67	67	67
20	S	90	90	65	90	56	56
21	T	85	85	90	85	80	80
22							

- We would like to know how many regions have seen a reduction in rainfall between 2010 and 2020

Example

```
1 import pandas as pd
2
3 # import the rainfall data
4 Rainfalldata = pd.read_csv("Rainfall.csv")
5 print(Rainfalldata.head())
6
7 # extract the columns for 2010 and 2020
8 Rainfall2010 = Rainfalldata['Rainfall 2010'].values
9 Rainfall2020 = Rainfalldata['Rainfall 2020'].values
```

Example

```
# How many regions have seen a decrease in rainfall
N = len(Rainfall2010)
count = 0
for i in range(0, N, 1):
    if Rainfall2020[i] < Rainfall2010[i]:
        count = count + 1

percentage_decreased = count/N*100

print('Of the {} regions, {} have reduced rainfall
    which is {:f6.2}% '.format(N, count,
    percentage_decreased))
```

Lecture summary

- blah
- blah
- what's next = functions