DATA SIENCE

Lecture 4

Lecture 4: Stochastic Thinking

Activate Windows
Go to Settings to activate Windows.

Newtonian Mechanics

- Every effect has a cause
- The world can be understood causally

1643 - 1727

Activate Windows
Go to Settings to activate Windows.

Copenhagen Doctrine

- Copenhagen Doctrine (Bohr and Heisenberg) of causal nondeterminism
  - At its most fundamental level, the behavior of the physical world cannot be predicted.
  - Fine to make statements of the form "x is highly likely to
  - occur," but not of the form "x is certain to occur."

Activate Windows
Go to Settings to activate Windows.

Stochastic Processes

- An ongoing process where the next state might depend on both the previous states and some random element

```
def rollDie():
    """returns an int between 1 and 6"""

def rollDie():
    """returns a randomly chosen int
       between 1 and 6"""
```

Activate Windows
Go to Settings to activate Windows.

Implementing a Random Process

```
import random

def rollDie():
    """returns a random int between 1 and 6"""
    return random.choice([1,2,3,4,5,6])

def testRoll(n = 10):
    result = ''
    for i in range(n):
        result = result + str(rollDie())
    print(result)
```

Activate Windows
Go to Settings to activate Windows.

Implementing a Random Process

```
import random

def rollDie():
    """returns a random int between 1 and 6"""
    return random.choice([1,2,3,4,5,6])

def testRoll(n = 10):
    result = ''
    for i in range(n):
        result = result + str(rollDie())
    print(result)
```


Activate Windows
Go to Settings to activate Windows.

Probability Is About Counting

- Count the number of possible events
- Count the number of events that have the property of interest
- Divide one by the other
- Probability of 11111?
  - 11111, 11112, 11113, …, 11121, 11122, …, 66666
  - 1/(6**5)

Activate Windows
Go to Settings to activate Windows.

Three Basic Facts About Probability

- Probabilities are always in the range 0 to 1. 0 if impossible, and 1 if guaranteed.
- If the probability of an event occurring is p, the probability of it not occurring must be $1 - p$
- When events are independent of each other, the probability of all of the events occurring is equal to a product of the probabilities of each of the events occurring.

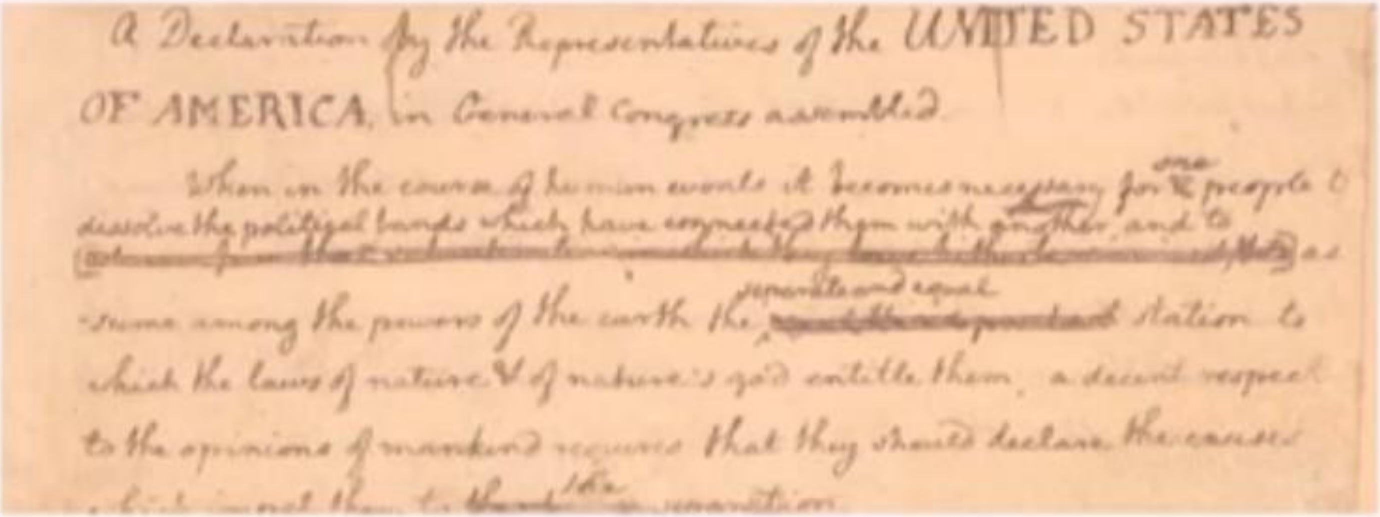
A : .5   B .4

A & B : .2

Activate Windows
Go to Settings to activate Windows.

Independence

- Two events are independent if the outcome of one event has no influence on the outcome of the other
- Independence should not be taken for granted



Activate Windows
Go to Settings to activate Windows.

Will One of the Patriots and Broncos Lose?

- Patriots have winning percentage of 7/8, Broncos of 6/8
- Probability of both winning next Sunday is 7/8 * 6/8 = 42/64
- Probability of at least one losing is 1 – 42/64 = 22/64

Activate Windows
Go to Settings to activate Windows.

A Simulation of Die Rolling

```
def runSim(goal, numTrials, txt):
    total = 0
    for i in range(numTrials):
        result = ''
        for j in range(len(goal)):
            result += str(rollDie())
        if result == goal:
            total += 1
    print('Actual probability of', txt, '=',
          round(1/(6**len(goal)), 8))
    estProbability = round(total/numTrials, 8)
    print('Estimated Probability of', txt, '=',
          round(estProbability, 8))

runSim('11111', 1000, '11111')
```

Activate Windows
Go to Settings to activate Windows.

```python
random.seed(0)

def runSim(goal, numTrials, txt):
    total = 0
    for i in range(numTrials):
        result = ''
        for j in range(len(goal)):
            result += str(rollDie())
        if result == goal:
            total += 1
    print('Actual probability of', txt, '=',
          round(1/(6**len(goal)), 8))
    estProbability = round(total/numTrials, 8)
    print('Estimated Probability of', txt, '=',
          round(estProbability, 8))

runSim('11111', 1000, '11111')

def sameDate(numPeople, numSame):
    possibleDates = range(366)
#    possibleDates = 4*list(range(0, 57)) + [58]\
```

```
TypeError: cannot
instantiate ctype
'EVP_MD_CTX' of unknown
size


In [5]: runfile('C:/Users/
John/Dropbox (MIT)/current/
mit/Teaching/600/
Fall16/6.0002/lecture4/
lecture 4/lecture4.py',
wdir='C:/Users/John/Dropbox
(MIT)/current/mit/Teaching/
600/Fall16/6.0002/lecture4/
lecture 4')
Actual probability of 11111
= 0.0001286
Estimated Probability of
11111 = 0.0

In [6]:
```

Activate Windows
Go to Settings to activate Windows.

Output of Simulation

- Actual probability = 0.0001286
- Estimated Probability = 0.0
- Actual probability = 0.0001286
- Estimated Probability = 0.0

pseudo
random

seed

- How did I know that this is what would get printed?

random.seed(0)

Activate Windows
Go to Settings to activate Windows.

```python

random.seed(0)

def runSim(goal, numTrials, txt):
    total = 0
    for i in range(numTrials):
        result = ''
        for j in range(len(goal)):
            result += str(rollDie())
        if result == goal:
            total += 1
    print('Actual probability of', txt, '=',
          round(1/(6**len(goal)), 8))
    estProbability = round(total/numTrials, 8)
    print('Estimated Probability of', txt, '=',
          round(estProbability, 8))

runSim('11111', 1000000, '11111')

def sameDate(numPeople, numSame):
    possibleDates = range(366)
#    possibleDates = 4*list(range(0, 57)) + [58]\
```

```
600/Fall16/6.0002/lecture4/
lecture 4')
Actual probability of 11111
= 0.0001286
Estimated Probability of
11111 = 0.0

In [6]: runfile('C:/Users/
John/Dropbox (MIT)/current/
mit/Teaching/600/
Fall16/6.0002/lecture4/
lecture 4/lecture4.py',
wdir='C:/Users/John/Dropbox
(MIT)/current/mit/Teaching/
600/Fall16/6.0002/lecture4/
lecture 4')
Actual probability of 11111
= 0.0001286
Estimated Probability of
11111 = 0.000128

In [7]:
```

Activate Windows
Go to Settings to activate Windows.

The Birthday Problem

- What's the probability of at least two people in a group having the same birthday
- If there are 367 people in the group?
- What about smaller numbers?
- If we assume that each birthdate is equally likely
  - $1-\frac{366!}{366^N*(366-N)!}$
- Without this assumption, VERY complicated

shoutkey.com/niece

Activate Windows
Go to Settings to activate Windows.

Approximating Using a Simulation

```
def sameDate(numPeople, numSame):
    possibleDates = range(366)
    birthdays = [0]*366
    for p in range(numPeople):
        birthDate = random.choice(possibleDates)
        birthdays[birthDate] += 1
    return max(birthdays) >= numSame
```

Activate Windows
Go to Settings to activate Windows.

Approximating Using a Simulation

```
def birthdayProb(numPeople, numSame, numTrials):
    numHits = 0
    for t in range(numTrials):
        if sameDate(numPeople, numSame):
            numHits += 1
    return numHits/numTrials

for numPeople in [10, 20, 40, 100]:
    print('For', numPeople,
          'est. prob. of a shared birthday is',
          birthdayProb(numPeople, 2, 10000))
    numerator = math.factorial(366)
    denom = (366**numPeople)*math.factorial(366-numPeople)
    print('Actual prob. for N = 100 =',
          1 - numerator/denom)
```

Activate Windows
Go to Settings to activate Windows.

```python
        birthdays[birthDate] += 1
    return max(birthdays) >= numSame

def birthdayProb(numPeople, numSame,
    numHits = 0
    for t in range(numTrials):
        if sameDate(numPeople, numSa
            numHits += 1
    return numHits/numTrials

import math

for numPeople in [10, 20, 40, 100]:
    print('For', numPeople,
          'est. prob. of a shared bi
          birthdayProb(numPeople, 2,
    numerator = math.factorial(366)
    denom = (366**numPeople)*math.fa
    print('Actual prob. for N = 100
          1 - numerator/denom)
```

```
current/mit/Teaching/600/Fall16/6.0002/lecture4/
lecture 4/lecture4.py', wdir='C:/Users/John/
Dropbox (MIT)/current/mit/Teaching/600/
Fall16/6.0002/lecture4/lecture 4')
Actual probability of 11111 = 0.0001286
Estimated Probability of 11111 = 0.000128

In [7]: runfile('C:/Users/John/Dropbox (MIT)/
current/mit/Teaching/600/Fall16/6.0002/lecture4/
lecture 4/lecture4.py', wdir='C:/Users/John/
Dropbox (MIT)/current/mit/Teaching/600/
Fall16/6.0002/lecture4/lecture 4')
For 10 est. prob. of a shared birthday is 0.1183
Actual prob. for N = 100 = 0.1166454118039999
For 20 est. prob. of a shared birthday is 0.4116
Actual prob. for N = 100 = 0.4105696370550831
For 40 est. prob. of a shared birthday is 0.8941
Actual prob. for N = 100 = 0.89054476188945
For 100 est. prob. of a shared birthday is 1.0
Actual prob. for N = 100 = 0.9999996784357714

In [8]:
```

Activate Windows
Go to Settings to activate Windows.

Why 3 Is Much Harder Mathematically

- For 2 the complementary problem is "all birthdays distinct"
- For 3 people, the complementary problem is a complicated disjunct
  - All birthdays distinct or
  - One pair and rest distinct or
  - Two pairs and rest distinct or
  - ...

Activate Windows
Go to Settings to activate Windows.

```python
        birthDate = random.choice(possibleDate
        birthdays[birthDate] += 1
    return max(birthdays) >= numSame

def birthdayProb(numPeople, numSame, numTrials
    numHits = 0
    for t in range(numTrials):
        if sameDate(numPeople, numSame):
            numHits += 1
    return numHits/numTrials

import math

for numPeople in [10, 20, 40, 100]:
    print('For', numPeople,
          'est. prob. of a shared birthday is'
          birthdayProb(numPeople, 3, 10000))
    numerator = math.factorial(366)
    denom = (366**numPeople)*math.factorial(36
    print('Actual prob. for N = 100 =',
          1 - numerator/denom)
```

```
Fall16/6.0002/lecture4/lecture 4/
lecture4.py', wdir='C:/Users/John/
Dropbox (MIT)/current/mit/Teaching/
600/Fall16/6.0002/lecture4/lecture 4')
For 10 est. prob. of a shared birthday
is 0.0011
Actual prob. for N = 100 =
0.1166454118039999
For 20 est. prob. of a shared birthday
is 0.0066
Actual prob. for N = 100 =
0.4105696370550831
For 40 est. prob. of a shared birthday
is 0.0651
Actual prob. for N = 100 =
0.89054476188945
For 100 est. prob. of a shared
birthday is 0.6359
Actual prob. for N = 100 =
0.9999996784357714

In [10]:
```

Activate Windows
Go to Settings to activate Windows.

Another Win for Simulation

- Adjusting analytic model a pain
- Adjusting simulation model easy

```
def sameDate(numPeople, numSame):
    possibleDates = 4*list(range(0, 57)) + [58]\
                    + 4*list(range(59, 366))\
                    + 4*list(range(180, 270))
    birthdays = [0]*366
    for p in range(numPeople):
        birthDate = random.choice(possibleDates)
        birthdays[birthDate] += 1
    return max(birthdays) >= numSame
```

Activate Windows
Go to Settings to activate Windows.

```python
            round(estProbability, 8))

#runSim('11111', 1000000, '11111')

def sameDate(numPeople, numSame):
#     possibleDates = range(366)
    possibleDates = 4*list(range(0, 57)) + [58
                    + 4*list(range(59, 366))\
                    + 4*list(range(180, 270))
    birthdays = [0]*366
    for p in range(numPeople):
        birthDate = random.choice(possibleDate
        birthdays[birthDate] += 1
    return max(birthdays) >= numSame

def birthdayProb(numPeople, numSame, numTrials
    numHits = 0
    for t in range(numTrials):
        if sameDate(numPeople, numSame):
            numHits += 1
    return numHits/numTrials

```

```
600/Fall16/6.0002/lecture4/lecture 4/
lecture4.py', wdir='C:/Users/John/
Dropbox (MIT)/current/mit/Teaching/
600/Fall16/6.0002/lecture4/lecture 4')
For 10 est. prob. of a shared birthday
is 0.0019
Actual prob. for N = 100 =
0.1166454118039999
For 20 est. prob. of a shared birthday
is 0.0097
Actual prob. for N = 100 =
0.4105696370550831
For 40 est. prob. of a shared birthday
is 0.0871
Actual prob. for N = 100 =
0.89054476188945
For 100 est. prob. of a shared
birthday is 0.7501
Actual prob. for N = 100 =
0.9999996784357714

In [11]:
```

Activate Windows
Go to Settings to activate Windows.

Simulation Models

- A description of computations that provide useful information about the possible behaviors of the system being modeled

Activate Windows
Go to Settings to activate Windows.