# ds-ua-112-hw01

September 20, 2019

## 0.1 Homework 1

### 0.1.1 Due Tuesday September 17, 11:59 PM

### 0.1.2 Collaboration Policy

While you may talk with others about the homework, we ask that you **write your solutions individually**. If you do discuss the assignments with others please **include their names** at the top of your notebook.

**Collaborators**: *list collaborators here*

### 0.1.3 Goal

Homework 1 will help prepare you for class along with the other assignments. The problems will review some of the programming that we will need throughout the semester. Please check the references for more information on unfamiliar material. Reach out to us in section and office hours for help.

**Tips** To learn about keyboard shortcuts, go to **Help -> Keyboard Shortcuts** in the menu above. Get help for a function by running a cell with the function name and a `?` at the end...you can escape by hitting `esc` several times. Alternatively type the function name, then - on your keyboard...you can press multiple times to show additional information.

### 0.1.4 Score Breakdown

| Question | Points |
|---|---|
| 1 | 2 |
| 2a | 1 |
| 2b | 1 |
| 2c | 2 |
| 2d | 2 |
| 3a | 3 |
| 3b | 3 |
| 4a | 2 |
| 4b | 2 |
| 4c | 3 |
| 4d | 3 |
| Total | 24 |

### 0.1.5 Question 1

Recall that summation (or sigma notation) is a way of expressing a long sum in a concise way. Let $a_1, a_2, ..., a_n \in \mathbb{R}$ and $x_1, x_2, ..., x_n \in \mathbb{R}$ be collections of real numbers. When you see $x_i$, you can think of the $i$ as an index for the $i^{th}$ $x$. For example $x_2$ is the second $x$ value in the list $x_1, x_2, ..., x_n$. We define sigma notation as follows:

$$\sum_{i=1}^{n} a_i x_i = a_1 x_1 + a_2 x_2 + ... + a_n x_n$$

We commonly use sigma notation to compactly write the definition of the arithmetic mean (commonly known as the `average`):

$$\bar{x} = \frac{1}{n}(x_1 + x_2 + ... + x_n) = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Recall the formula for population variance below:

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}$$

Complete the functions below to compute the population variance of `population`, an array of numbers. For this question, do not use built in NumPy functions; we will use NumPy to verify your code.

[5]:
```
import numpy as np
```

[6]:
```
def mean(population):
    """
    Returns the mean of population (mu)

    Keyword arguments:
    population -- a numpy array of numbers
    """
    sum = 0
    for i in range(len(population)):
        sum = sum + population[i]
    mean = sum/len(population)
    return mean

def variance(population):
    """
    Returns the variance of population (sigma squared)

    Keyword arguments:
    population -- a numpy array of numbers
    """
    sum = 0
    for i in range(len(population)):
        sum = sum + ((population[i] - mean(population))**2)
    variance = sum/len(population)
    return variance
```

Run the following cell to test your function against `numpy`

[7]:
```
# TEST
population_0 = np.random.randn(100)
np.isclose(mean(population_0), np.mean(population_0), atol=1e-6)
```

[7]:
```
True
```

[8]:
```
# TEST
population_0 = np.random.randn(100)
np.isclose(variance(population_0), np.var(population_0), atol=1e-6)
```

[8]:
```
True
```

### 0.1.6 Question 2

Suppose you're trying to meet your friend in either Central Park (denoted C) or Riverside Park (denoted R).

Right now, you know that the chances of your friend being at $z$ are

| | Chance |
|---|---|
| **z = Central Park** | 0.6 |
| **z = Riverside Park** | 0.4 |

a. Use `numpy` to create a $2 \times 1$ array called `v` with chance of being in Central Park and Riverside Park. Try testing the dimensions.

[9]:
```
v = np.array([[.6], [.4]])

# TEST
v.shape
```

[9]:
```
(2, 1)
```

Since your friend like to wander around, you know that after 1 hour the chances of her walking from $x$ to $y$ are

| | x = Central Park | x= Riverside Park |
|---|---|---|
| **y = Central Park** | 0.6 | 0.7 |
| **y = Riverside Park** | 0.4 | 0.3 |

b. Use `numpy` to create a $2 \times 2$ matrix `M` with chance of walking between parks. Try testing the dimensions.

[10]:
```
M = np.array([[.6, .7], [.4, .3]])
M.shape
```

[10]:
```
(2, 2)
```

c. Use the `numpy` sum method to sum the entries over columns. Do the entries sum to 1 -- why?

Use the numpy sum method to sum the entries over rows. Do the entries sum to 1 -- why?

[13]:
```
for j in reversed(range (2)):
    print(M.sum(axis = j))
```

```
[1.3 0.7]
[1. 1.]
```

Explain Why

The columns are where the friend goes given X = x, so the entire sample space is within the column. The rows are the probabilities of where the friend will go, and there is no given which already occured. The sample space would be Mv, using Law of Total Probability.

d. Use the Law of Total Probability to determine the likelihood that your friend is in Central Park or Riverside Park after 1 hour.

**Hint: Use the @ symbol for multiplication**

[27]:
```
#create a new 2x1 array s such that Mv = f
#f is probabilities of being in C or R after one hour
f = np.dot(M,v)
f = M@v
f = np.matmul(M,v)
print(f)
```

```
[[0.64]
 [0.36]]
```

### 0.1.7 Question 3

a. Debugging is jargon for inspecting the code in a program line by line for errors. Try inspecting the following code -- locate, document and correct each error.

[32]:
```
# Write a program that will average 3 numeric exam grades, return an average␣
 ↪test score, a corresponding letter grade, and a message stating whether the␣
 ↪student is passing.

# Average         Grade
# 90+         A
# 80-89         B
# 70-79         C
# 60-69         D
# 0-59         F

exam_one = int(input("Input exam grade one: "))

exam_two = int(input("Input exam grade two: "))
#missing int
```

```
exam_three = int(input("Input exam grade three: "))
#exam_three not exam_3
#int not string

grades = [exam_one, exam_two, exam_three]
#commas
sum = 0
for grade in grades:
    #list is called grades not grade
    sum = sum + grade
    #not properly indented, tabs over spaces any day

avg = sum / len(grades)
#grades not grdes

if avg >= 90:
    letter_grade = "A"
elif avg >= 80 and avg < 90:
#missing colon
    letter_grade = "B"
elif avg > 69 and avg < 80:
    letter_grade = "C"
    #quotes
elif avg <= 69 and avg >= 65:
    letter_grade = "D"
else:
    #else not elif
    letter_grade = "F"

for grade in grades:
    print("Exam: " + str(grade))

    print("Average: " + str(avg))

    print("Grade: " + letter_grade)

if letter_grade is "F":
    #letter_grade not letter-grade
    print ("Student is failing.")
    #parentheses
else:
    print ("Student is passing.")
    #parentheses
```

```
Input exam grade one: 32
Input exam grade two: 34
Input exam grade three: 43
```

```
Exam: 32
Average: 36.333333333333336
Grade: F
Exam: 34
Average: 36.333333333333336
Grade: F
Exam: 43
Average: 36.333333333333336
Grade: F
Student is failing.
```

[ ]:
```
# TEST 1
# Exams: 89, 90, 90
# Average: 90
# Grade: A
# Student is passing.

# TEST 2
# Exams: 50, 51, 0
# Average: 33
# Grade: F
# Student is failing.
```

b. We can use debugging tool to explore the state of a running program without using print() statements everywhere. In Jupyter notebooks, we can import the `pdb` package for the debugger.

[33]:
```
import pdb
```

We can use the debugger in two ways:

1. After the code threw an exception, we work **backwards** from the problem.

- Run `pdb.pm()` to start the debugger after an exception.
- Use the `up` command to work backwards.
- Use `quit` to quit the debuggger.

2. We place a **break-point** where execution halts and we work **forwards** line by line.

- Place `pdb.set_trace()` in the line where execution should halt.
- Try using the following to work forwards.

    1. `list` - Displays 11 lines around the current line or continue the previous listing.
    2. `step` - Execute the current line, stop at the first possible occasion.
    3. `next` - Continue execution until the next line in the current function is reached or it returns.
    4. `break` - Set another breakpoint at a line (depending on the argument provided).
    5. `return` - Continue execution until the current function returns.

- For more information type `help`

Save the following script to a file called `fib.py`.

```
# fib.py

import sys

def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)

for i in range(len(sys.argv)):
    arg = sys.argv[i]
    num = int(arg)
    print("fib of", num, " = ", fib(num))
```

Try executing the following in a cell

[4]:
```
run fib.py 4 5 6
```

```
fib of 4  =  3
fib of 5  =  5
fib of 6  =  8
```

What went wrong...use `pdb` to locate, document and correct the errors in `fib.py`.

[ ]:
```
#need to start from index = 1
import sys

def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)

for i in range(1, len(sys.argv)):
    arg = sys.argv[i]
    num = int(arg)
    print("fib of", num, " = ", fib(num))
```

### 0.1.8 Question 4

Consider the following scenario:

Only 1% of 40-year-old women who participate in a routine mammography test have breast cancer. 80% of women who have breast cancer will test positive, but 9.6% of women who don't have breast cancer will also get positive tests.

a. Fill in the conditional probability table

| | x = Has Cancer | x= Does Not Have Cancer |
|---|---|---|
| **Tested Positive given x** | .8 | .096 |
| **Tested Negative given x** | .2 | .904 |

**SOLUTION:**

---

b. Fill in the joint probability table

| | Has Cancer | Does Not Have Cancer |
|---|---|---|
| **Tested Positive** | .008 | .09504 |
| **Tested Negative** | .002 | .89496 |

**SOLUTION:**

---

c. Suppose we know that a woman of this age tested positive in a routine screening. What is the probability that she actually has breast cancer?

**Hint:** Use Bayes' rule.
**SOLUTION:** P(cancer | positive) = [P(positive | cancer)P(cancer)]/[P(positive | cancer)P(cancer) + P(positive | cancer')P(cancer')];
P(cancer | positive) = (.8 * .01)/(.8 * .1 + .096 * .01)
P(cancer | positive) = .008/(.008 + .09504)
P(cancer | positive) = .008/.10304
P(cancer | positive) = .077639

---

d. Sometimes, 40-year-old men will also get mammographies, if they are suspected to be at high risk to breast cancer. The overall prevalence of breast cancer in men who participate in the test is lower (0.01%). As with women, 80% of men who have breast cancer will test positive. Due to physiological diferences, only 2.4% of men who don't have breast cancer will test positively.

For each of the following pairs of events $A$ and $B$, please determine whether or not they are independent:

1. $A$ = the subject's sex (man or woman), $B$ = having breast cancer.
2. $A$ = the subject's sex, $B$ = testing positively for breast cancer given having breast cancer.
3. $A$ = the subject's sex, $B$ = testing positively for breast cancer.

**Hint:** Events $A$ and $B$ are independent if $P(B \mid A) = P(B)$
**SOLUTION:** (Let A be event of female)

1. P(B) = P(B | A)P(A) + P(B | A')P(A') = .050005

   P(A) = .5
   P(B)P(A) = .0250025
   P(A and B) = P(B | A)P(A) = .4 =/= .0250025
   NOT INDEPENDENT

2. P(B and A) = P(B)P(A) = .8

   INDEPENDENT

3. P(B) = .0635588

   P(A)P(B) = .0317794
   P(B and A) = .10304

### 0.2 NOT INDEPENDENT

[ ]: