LS30A - Lab 3

Aaron

Review about modeling

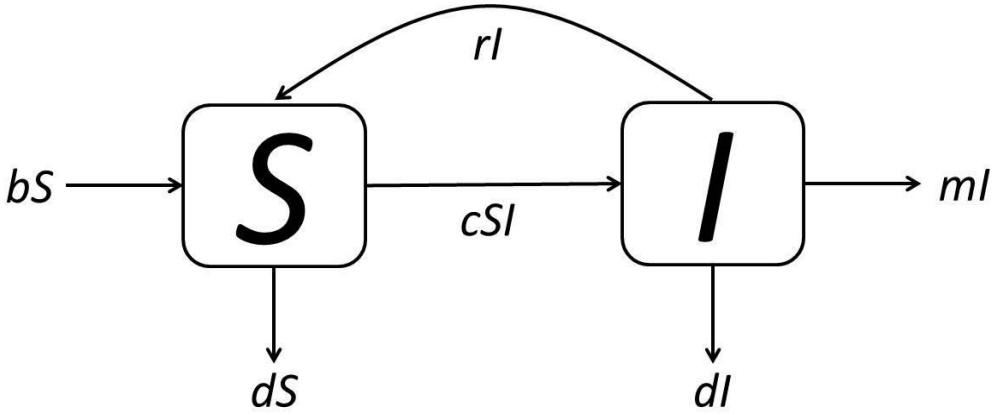
1. **For the mathematical model below,**

a. Identify the state variable(s).
b. Sketch a flow/block/box diagram.
c. Write a differential equation model.
d. Sketch the axes for a trajectory graph and time series graph for this model.

(From Garfinkel Midterm W16) Mountain lions in the Santa Monica Mountains are affected by mange parasites. Using the following assumptions to write a model for the susceptible and infected mountain lion populations.

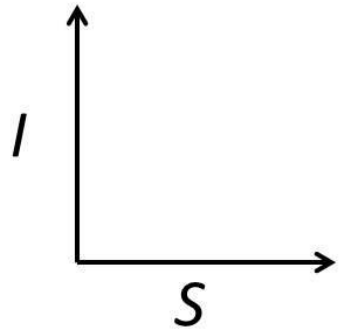
- **Susceptible individuals are born at a per-capita rate *b*.**
- **Both susceptible and infected individuals die from accidents, old age, etc. at per-capita rate *d*.**
- **Infected individuals die from mange at per-capita rate *m*.**
- **When a susceptible individual meets an infected one, the probability of it becoming infected is *c*.**
- **Infected individuals recover at per-capita rate *r*.**

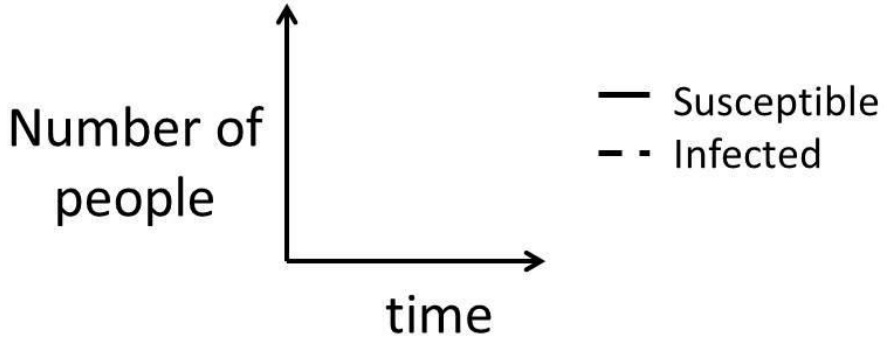
Hint: (Let S = # of susceptible mountain lions, I = # of infected mountain lions)



$S' = bS - dS - cSI + rI$

$I' = -dI - mI + cSI - rI$





Review about Lab1 & Lab2

Write a function that takes a list of numbers and outputs a list whose elements are double those of its input list. The function should do this without the number of elements in the list being known ahead of time or given as an input. For example, `double([1,2,3])` **should return** `[2,4,6]`**.**

```python
def double_list(list_nums):
    # Purpose: outputs a list whose elements are double those of its input list
    # Input: list of numbers
    # Output: list of numbers (whose elements are double those of its input list)

    doubles = [] # Create new list called doubles

    for item in list_nums:
        # implement code inside for loop for item using each value of list_nums
        double_item = 2*item
        # Assign 2*item to variable double_item
        doubles.append(double_item)
        # Add double_item to end of list doubles

    return doubles # Output double

double_list([1,5,-3,8.1]) # Test function with variety of input
```

```
[2, 10, -6, 16.2000000000000]
```

1. Writing functions directly

```
g(x) = x + 5
g(1)
```

```
6
```

2. Making lists automatically

srange(m)

srange(m, n)

srange(m, n, step)

```
srange(10)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
srange(1, 10)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
srange(1,3,0.5)
```

```
[1.00000000000000, 1.50000000000000, 2.00000000000000, 2.50000000000000]
```

3. Iteration

```
#Define the function we want to iterate
f(x) = 2*x
#Create list of numbers defining how many times to iterate
count = srange(0,10)
#val is the variable that will hold our intermediate
#calculations.  We start with val being 3.
val=3
for i in count:
    #Plug val into f and assign the result to val.
    a = f(val)
    val = a
#Show the final result
val
```

```
3072
```

2^10 * 3 = 3072

3. From script to Function

```
1
2  def iterate_2x(initial_value, num_times):
3      f(x) = 2*x
4      count = srange(0, num_times)
5      val = initial_value
6      for i in count:
7          a = f(val)
8          val = a
9      return val
10
11 iterate_2x(3, 10)
```

```
3072
```

Input: 1. initial value
2. Iteration time
Output = 2^(iteration time) * initial value