Generic
Python for Ecologists

Tao Hong, Tom Purucker, Chance Pascale

Ecological Society of America Workshop
Portland, OR
hongtao510@gmail.com
August 1st, 2012

Population models in Übertool

1. Exponential
2. Logistic
3. Gompertz
4. Fox Surplus yield
5. Maximum Sustainable Yield
6. Yule-Furry
7. Feller-Arley
8. Leslie

Exponential Model

- Mathematical equation

$$N_t = N_0 e^{rt}$$

  - N_0 is the initial number of individuals
  - N_t is population size at t
  - r is intrinsic growth rate
  - t is duration
- We have three inputs and one output

Code in Python (exp)

- Define a function

```
def exponentialgrow(N_o , r,  T):
    index_set = np.arange(T+1)        #How to do this in np.linspace?
                                      #index_set = np.linspace(0,T,T+1)
    x = np.zeros(len(index_set))      #create an array to hold the results
    x[0] = N_o                        #initial condition
    for t in index_set[1:]:           #t starts at 0, ends at T
        x[t] = N_o*np.exp(r*t)
    return x
```

- Call the defined function

```
>>>N_t=exponentialgrow(10, 0.4, 10)
>>>[  10.         14.91824698   22.25540928   33.20116923   49.53032424
    73.89056099  110.23176381  164.44646771  245.32530197  365.98234444
    545.98150033]
```

It is your turn now!

- Please code the logistic population model

$$N_t = N_{t-1} + r_0 N_{t-1}(1 - \frac{N_{t-1}}{K})$$

  - N_t is population size at t
  - N_{t-1} is population size at t-1
  - K is population capacity
  - r_0 is max growth rate
  - t is simulation duration
- We have four inputs and one output

Code in Python (logistic)

- Define a function

```
def logisticgrow(N_o, T, r, K):
    index_set = np.arange(T+1)
    x = np.zeros(len(index_set))
x[0] = N_o
    for t in index_set[1:]:
        x[t] = x[t-1] + (r)*x[t-1]*(1 - x[t-1]/float(K))
return x
```

Why use float(K)?
Try 10/100
And 10/float(100)

- Call the defined function

```
>>>N_t=logisticgrow(10, 10, 0.4, 100)
>>>[ 10.          13.6         18.30016     24.28064058  31.63469878
 40.28556162  49.90808037  59.90804657  69.51536903  77.99197051
 84.85776886]
```

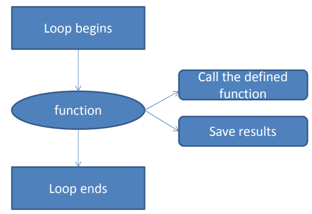
Feller-Arley (birth-death) Markov Process

- No internal population structure and each individual can give birth and death with constant rates

- N_t=N_{t-1}-1+N_{birth}-N_{death}
- N_{birth} ~ binomial (N, P_{birth})
- N_{death} ~ binomial (N, P_{death})
- Let's look at the code 'population_modeling_bdp.py'

Monte Carlo Simulation

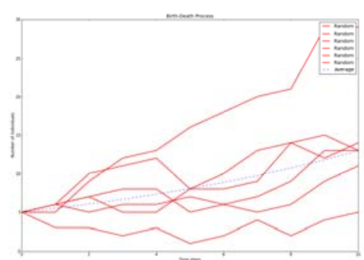
- Wrap the code by a loop
- Save results of each iteration



- p

Plot Results

- matplotlib, a library to illustrate your data



Leslie Model

$$\begin{bmatrix} n_0 \\ n_1 \\ \vdots \\ n_k \end{bmatrix}_{t+1} = \begin{bmatrix} F_o & F_1 & \cdots & F_k \\ S_0 & 0 & \cdots & 0 \\ 0 & \ddots & \cdots & 0 \\ 0 & 0 & S_{k-1} & 0 \end{bmatrix} \begin{bmatrix} n_0 \\ n_1 \\ \vdots \\ n_k \end{bmatrix}_t$$

```
for i in range(0, T):
    n=np.dot(l_m, n_o)
    n_o=n
    n_f[:,i]=n.squeeze()
```

Why squeeze?
Try n.ndim and
n.squeeze().ndim

Let's look at the script 'population_modeling_lm.py'

10