INF 674 S1: Galton-Watson Process

Céline Comte & Fabien Mathieu

2017-2018

To start the course, we propose to investigate the **Galton-Watson process**. This process was introduced to depict the propagation of some feature (family surname, DNA) through generations, and in particular to estimate its probability of extinction.

A Galton-Watson process can be represented as a (directed) random tree. It is built generation by generation as follows. At generation 0, there is a single individual, the ancestor of the population, represented by the tree *root*. Starting from this, each individual from a given generation i gives birth to a certain number of individual at generation $i+1$, represented by its direct successors in the tree.

The unique parameter of the process is the distribution of the number of children of each node. Specifically, we assume that the number of children of a node is independent from the other nodes, drawn according to some given distribution $(p_k)_{k \in \mathbb{N}}$ (s.t. $\sum_{k=0}^{\infty} p_k = 1$). We let μ denote the average number of children, supposed finite:

$$\mu = \sum_{k=0}^{\infty} k p_k < +\infty.$$

Note that there are multiple ways to explore the tree. For example:

- **Generation by generation**: This is the view we presented above. We call G_i the random variable which counts the number of nodes at generation i.
- **Active node by active node**: The nodes are visited one by one to decide on their number of offsprings. We keep track of the number of nodes which are *active* in the sense that we have discovered them but we have not drawn their number of offsprings yet. As long as there are active nodes, we can perform a *termination* which consists in desactivating a node, drawing the number of its children according to $(p_k)_{k \in \mathbb{N}}$ and adding these children (if any) to the set of active nodes. We call X_t the number of active nodes after t terminations, with the convention that $X_0 = 1$. Observe that $(X_t)_{t \in \mathbb{N}}$ defines a Markov process which is similar to a birth-and-death process, except that state 0 is absorbing.

The goal of the practical is to play with the two views to understand Galton-Watson processes.

If you want to deepen your theoretical knowledge of this process, you can read Chapter 1 from the book Epidemics and Rumours in Complex Networks (https://www.cambridge.org/core/books/epidemics-and-rumours-in-complex-networks/8C1D162F44C2C09F2B913038A7FA8BF6) (which is **not** mandatory).

In [ ]:
```
%pylab inline
```

1. Bimodal distribution

We first assume a very simple children distribution, the *bimodal* distribution, where a node can only have 0 or 2 children: $p_0 = 1-\mu/2$, $p_2 = \mu/2$, $p_k = 0$ for $k \notin \{0, 2\}$. In this section, we focus on values of the mean μ which range between 0 and 2, and perform an empirical study of the associated Galton-Watson process.

Note: Try to write a flexible code, as parameters and distributions will change.

Question 1

We first consider the *generation by generation* exploration. Write a function `generation_growth` that returns the values G_i observed during a realization of the process (up to generation `imax`).

Run it a few times with different values of the mean μ. Can you comment?

The function `random.rand` of `numpy` package may be handy.

Code:

In [ ]:
```
def generation_growth(μ, imax = 20):
    g = ones(imax + 1, dtype = int)
    # to be completed
    return g
```

Discussion:

In [ ]:
```
μ = 1.
generation_growth(μ)
```

Question 2

We now consider the *active node by active node* exploration. Write a function `active_growth` that returns the values X_t observed during a realization of the process (fix a maximal number `tmax` of terminations). Warning: your function should not return negative values.

Run it a few timess with different values of the mean μ. Can you comment?

Code:

In [ ]:
```
def active_growth(μ, tmax = 20):
    x = zeros(tmax + 1, dtype = int)
    # to be completed
    return x
```

Discussion:

In [ ]:
```
μ = 1.
active_growth(μ)
```

Question 3

Write two functions `estimate_generation` and `estimate_active` that estimate $\mathbb{E}(G_i)$ and $\mathbb{E}(X_t)$ by averaging the results over n runs.

For $\mu = 1/2$ and $\mu = 4/3$, display the results in figures and comment.

Code:

In [ ]:
```
def estimate_generation(μ, imax = 10, n = 1000):
    g = zeros(imax+1)
    # to be completed
    return g / n
```

In [ ]:
```
def estimate_active(μ, tmax = 100, n = 1000):
    x = np.zeros(tmax+1)
    # to be completed
    return x/n
```

Discussion:

In [ ]:
```
μ = 4/3
```

In [ ]:
```
imax = 10

figure()
semilogy(arange(imax+1), estimate_generation(μ, imax), label='Estimate of $\mathbb{E}(G_i)$')
xlabel('Generation')
ylabel('Number of individuals')
legend()
show()
```

In [ ]:
```
tmax = 100

figure()
plot(arange(tmax+1), estimate_active(μ, tmax), label='Estimate of $\mathbb{E}(X_t)$')
xlabel('Step')
ylabel('Number of individuals')
legend(loc=2)
show()
```

Question 4

At each realization, you may face extinction in the sense that none of the nodes of this generation has children. Write a function `estimate_extinction` that uses the function `active_growth` of Question 2 to estimate the probability of extinction P_{ext}. Run it on a few values of μ.

Code:

Question 5 (Bonus)

Evaluate $\mathbb{E}(G_i)$ conditioned on the run has lead to extinction or not. Discuss the results.

Answer:

2. Extinction

We focus now on the probability P_{ext} of extinction, which is the probability that the total population is finite. We will observe experimentally the following phase transition:

- If $\mu<1$, then $P_{ext} = 1$.
- If $\mu>1$, then $P_{ext}<1$.

The proof of this result is given in Epidemics and Rumours in Complex Networks (https://www.cambridge.org/core/books/epidemics-and-rumours-in-complex-networks/8C1D162F44C2C09F2B913038A7FA8BF6), along with more details (in particular, the behavior of the process when $\mu = 1$).

Question 1

Give an equality that relates P_{ext} and $(p_k)_{k \in \mathbb{N}}$.

Answer:

Question 2

We consider the bimodal distribution of Exercice 1. Admitting that P_{ext} is the smallest solution of the previous equation in the interval $[0, 1]$, relate P_{ext} and μ. Write a (very) small function `pext_bim_exact` that computes P_{ext} for a list of μ's.

Answer:

Question 3

Adapt the function from Question 4 of Exercice 1 to estimate P_{ext} by simulation for multiple values of μ. Suggested values: `t=10`, `t=100`, `t=1000`, `n=1000`, μ = `np.arange(0,2.05,.05)`. Display the results you obtain against the theoretical value obtained in the previous question. Discuss the difference.

Code:

Discussion:

Question 4

Evaluating the results by simulation has an inherent lack of accuracy. Try to compute exactly the probability that all nodes are dead after t terminations. Display the results and compare.

Hint: for $t<\infty$, write a function `pop_after_t` that computes the **distribution** of the number of active nodes after t terminations as a function of $p = (p_k)_{k \in \mathbb{N}}$. The function `convolve` from `numpy` package may be handy.

Answer:

Code:

3. Other Distributions

Question 1

We now consider a geometric distribution $p_k = (1 - a)a^k$, or each $k \in \mathbb{N}$, for some $0 \leq a<1$. Relate a and μ and study the extinction probability like you did for the bimodal case. In particular, give the equation P_{ext} should verify. Compute P_{ext} as a function of μ for $\mu \in [0, 2]$. For the non trivial cases, you can use an iterative computation of the solution. To validate the result, you should for instance:

- Adapt `pop_after_t` to compute P_{ext} after `t=1000` terminations for a truncated geometric distribution.
- Run multiple simulations using a geometric generator. The function `random.geometric` from `numpy` package may be handy.

Display the results.

Code:

Question 2