# Birth-Death Process

Jose Manuel de Frutos

15th August 2022

# Index

# Birth-Death Process

- Simple model of population dynamics called a birth-death processes. Population fluctuations (birth and death rates, $\beta_{birth}$ and $\beta_{death}$) depend only on the population size, $n$.

- Lets consider the following situation: protein synthesis. The rate at which the population increases depends only on the presence of the ingredients and is thus is fixed and a clearance rate (deaths) that is proportional to the total population, $n$.

- Two possible events: births (or synthesis) and deaths (or decays).

$$\beta_{birth} = (constant)$$
$$\beta_{death} = (k_0) * n$$

- $\beta_{birth}$ and $(k_0)$ are parameters of the model that would, in principle, be determined by experimental measurements. Both rates have units of [events/time].

# BIRTH-DEATH PROCESS

- Deterministically, the population size $n$ can be described by the ordinary differential equation:

$$\frac{dn(t)}{dt} = b - b \cdot n(t)$$

- Solving this differential equation we get:

$$n(t) = c \cdot e^{-d \cdot t} + \beta/d$$

Where $c = n(0) - \beta/d$.

# EXTREME SOLUTIONS

Simplest cases of pure birth process and pure death process.

- **pure birth process**, the governing equation is: $n(t) = b \cdot t + n(0)$. So any numerical solution that we obtain has to follow a linear pattern of $\beta$ slope, for this case.
- **pure death process**, the governing equation is: $n(t) = e^{-d \cdot t} + n(0) - 1$. In this case the decreasing exponential pattern has to be observed.
- General equation, we obtain the asymptotic trend. With previous solution for the general problem: $n = \beta/d$. The concavity will depend on the value of $C$.

# Index

# Stochastic Simulation Algorithm

- Two possible events that might occur and these might each happen at different rates: birth and death.

$$\frac{P_{death}}{P_{birth} + P_{death}} = \frac{d \cdot N}{\beta + d \cdot N}, \qquad \frac{P_{birth}}{Pdeath + P_{birth}} = \frac{\beta}{\beta + d \cdot N}$$

- **Bernoulli Trial**. Event can either be a birth or a death, need to be able determine which event actually occured. We choose a random number between $(0,1)$ and see if it is less than $\frac{d \cdot N}{\beta + d \cdot N}$ or bigger. In each case we add or subtract an individual to the population (Monte-Carlo).

- For simplicity and to avoid divisions, we take a random number between $(0, d \cdot N)$. And we check that it is less than or greater than $d \cdot N$.

- Events in birth-death processes do not occur at regular time intervals; instead we can only say that the odds of an event occurring within any small time interval $\Delta t$ is equal to $a_0\Delta t$, where is the mean rate of event occurrences. A sequence of random wait times is an example of what is called a "Poisson Process."

- Wait-times between sequential events follow an exponential distribution where the probability of the next event occurring after a duration $\tau$ (give-or-take $\Delta t$) is given by:

$$P(\tau) = a_0 \cdot e^{-a_0 \cdot \tau}$$

- By **superposition property** we get that $\tau = -\frac{\log r}{\beta + d \cdot x}$. Where $r$ is a random number between $(0, 1)$.
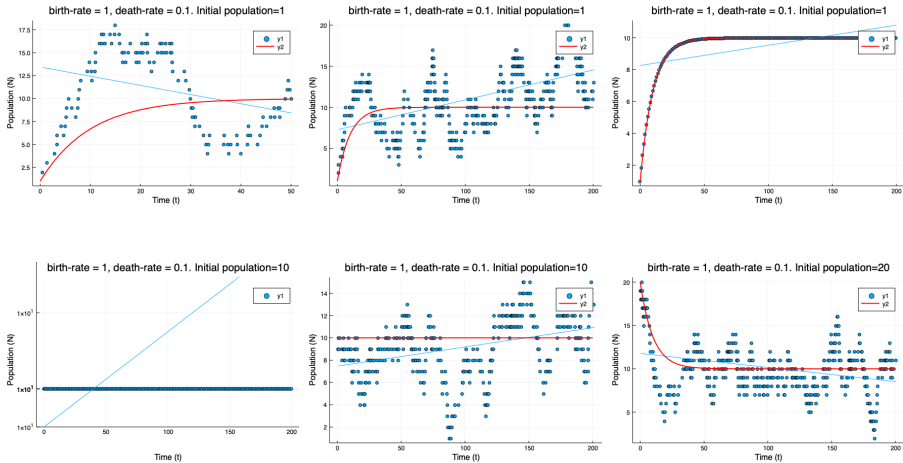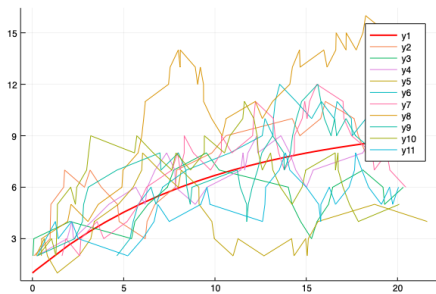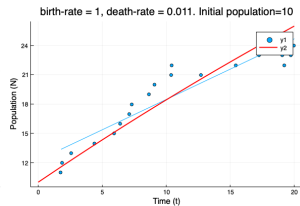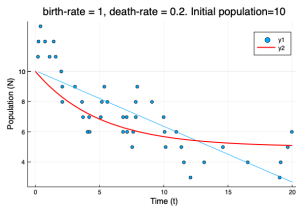
FIGURE: Some examples of simulation
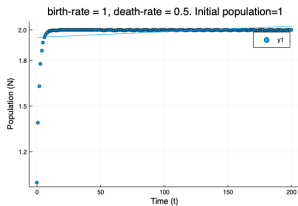
Birth-rate = 1, death-rate = 0.5. Initial population=1 · Birth-rate = 1, death-rate = 0.2. Initial population=10 · Birth-rate = 1, death-rate = 0.011. Initial population=10

# Index

We have divided the package as follows:

- We have a file **Utils.jl** where we have the utility functions. Mainly we have the functionality to generate random numbers.
- **DefaultParameters.jl** where we have the structure to define the parameters $\beta$ and $d$ of the model.
- **Gillespie.jl** and **RungeKutta.jl** here we define the simulation algorithms.
- We have separated the visualization and the obtaining of statistics in independent files. **Visualization.jl** and **Stats.jl**.
- We expose the functions in the file **BirthDeathProcess.jl**.

Finally we have implemented tests and a documentation in html. These are located in the tests and doc folder.

We have also implemented a github ci workflow. Find it ci.yml.

- Compiles the package for different versions of julia and in mac, windows and ubuntu environments.
- Run the package tests. We have enabled a code coverage manager (codecov).
- Generate the documentation. Created with Documenter.jl

Daniel T Gillespie et al.
Stochastic simulation of chemical kinetics.
*Annual review of physical chemistry*, 58(1):35–55, 2007.

Nicolas Lanchier.
*Stochastic modeling*.
Springer, 2017.