Simulating reaction systems with the Gillespie algorithm

Nico Gort Freitas // MCB111 11/11/22

Recap on reaction system

r	reaction	propensity	η(R)	Description
1	$Gene(I) \xrightarrow{k_b} Gene(A)$	$k_b \times (1 - \mathbf{1}_A)$	0	Gene activation
2	$Gene(A) \xrightarrow{k_u} Gene(I)$	$k_u \times \mathbf{1}_A$	0	Gene inactivation
3	$Gene(A) \xrightarrow{k_1} Gene(A) + RNA$	$k_1 \times \mathbf{1}_A$	+1	RNA synthesis
4	$RNA \xrightarrow{k_2} \emptyset$	$k_2 R$	-1	RNA degradation

How do we make sure no transcription is modeled to occur when the gene is inactivated?

Use an indicator function: $\mathbf{1}_A = \begin{cases} 1 & \text{if gene is activated} \\ 0 & \text{otherwise} \end{cases}$

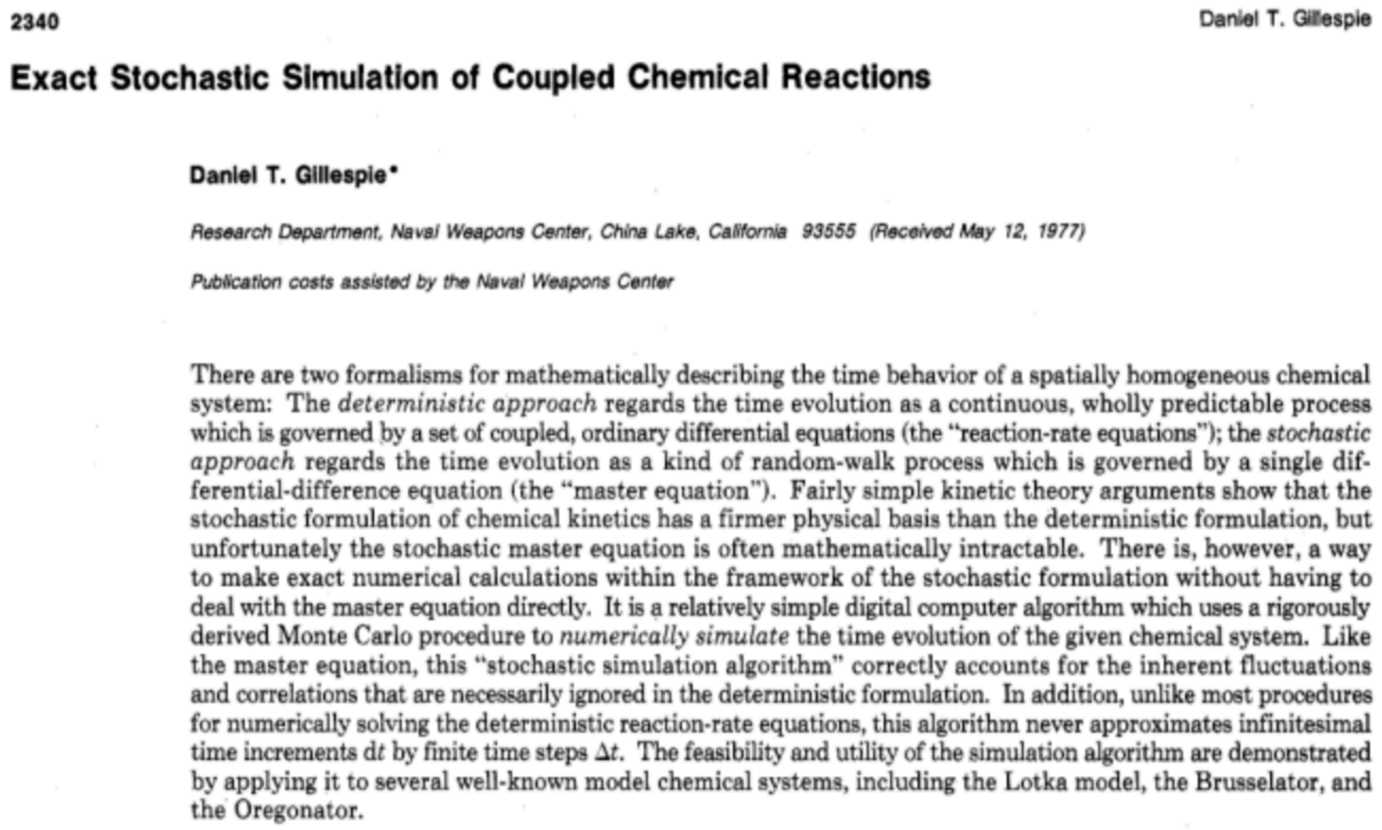
Naive simulation of reaction system

- Choose a short enough step size to avoid simultaneous reactions
- At each t+Δt:
    - Compute reaction probabilities given Δt
    - Sample whether any and which reaction occurs
    - Update abundances and rates

For low molecular numbers (and therefore infrequent collisions), no reaction would occur on most steps

What if we could skip straight to when the next reaction occurs, instead of simulating endless infinitesimal steps?

Gillespie Stochastic Simulation Algorithm



2340 Daniel T. Gillespie

Exact Stochastic Simulation of Coupled Chemical Reactions

Daniel T. Gillespie*

Research Department, Naval Weapons Center, China Lake, California 93555 (Received May 12, 1977)

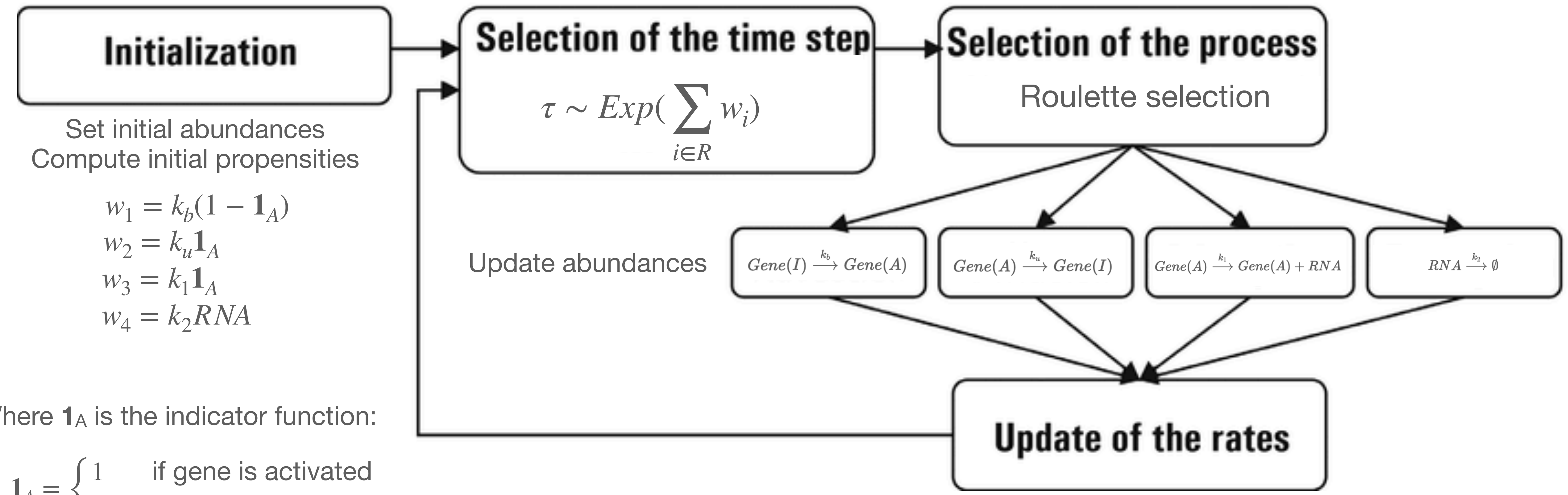
Publication costs assisted by the Naval Weapons Center

There are two formalisms for mathematically describing the time behavior of a spatially homogeneous chemical system: The *deterministic approach* regards the time evolution as a continuous, wholly predictable process which is governed by a set of coupled, ordinary differential equations (the "reaction-rate equations"); the *stochastic approach* regards the time evolution as a kind of random-walk process which is governed by a single differential-difference equation (the "master equation"). Fairly simple kinetic theory arguments show that the stochastic formulation of chemical kinetics has a firmer physical basis than the deterministic formulation, but unfortunately the stochastic master equation is often mathematically intractable. There is, however, a way to make exact numerical calculations within the framework of the stochastic formulation without having to deal with the master equation directly. It is a relatively simple digital computer algorithm which uses a rigorously derived Monte Carlo procedure to *numerically simulate* the time evolution of the given chemical system. Like the master equation, this "stochastic simulation algorithm" correctly accounts for the inherent fluctuations and correlations that are necessarily ignored in the deterministic formulation. In addition, unlike most procedures for numerically solving the deterministic reaction-rate equations, this algorithm never approximates infinitesimal time increments dt by finite time steps Δt. The feasibility and utility of the simulation algorithm are demonstrated by applying it to several well-known model chemical systems, including the Lotka model, the Brusselator, and the Oregonator.

DELTA
BETTER FLIGHTS ONE CUP AT A TIME
FOR THE CHANCE TO WIN A ROUND TRIP TICKET, TAG
#DeltaCupContest

- Samples exact solutions to the master equation
- Doesn't have to simulate infinitesimal Δt

Gillespie SSA workflow


Initialization
Selection of the time step
$\tau \sim Exp(\sum_{i \in R} w_i)$
Selection of the process
Roulette selection
Update abundances
$Gene(I) \xrightarrow{k_b} Gene(A)$
$Gene(A) \xrightarrow{k_u} Gene(I)$
$Gene(A) \xrightarrow{k_1} Gene(A) + RNA$
$RNA \xrightarrow{k_2} \emptyset$
Update of the rates

Set initial abundances
Compute initial propensities

$$w_1 = k_b(1 - \mathbf{1}_A)$$
$$w_2 = k_u \mathbf{1}_A$$
$$w_3 = k_1 \mathbf{1}_A$$
$$w_4 = k_2 RNA$$

Where $\mathbf{1}_A$ is the indicator function:

$$\mathbf{1}_A = \begin{cases} 1 & \text{if gene is activated} \\ 0 & \text{otherwise} \end{cases}$$

$$w_1 = k_b(1 - \mathbf{1}_A)$$
$$w_2 = k_u \mathbf{1}_A$$
$$w_3 = k_1 \mathbf{1}_A$$
$$w_4 = k_2 RNA$$

Exponentially distributed waiting times

We sample our waiting times from an exponential distribution with a rate ⟨1/τ⟩ equal to the sum of propensities

$$\tau \sim Exp(\sum_{i \in R} w_i) \equiv Exp(w_R)$$

Exponential PDF:

$$P(\tau \,|\, w_R) = w_R e^{-\tau w_R}$$

Exponential CDF and associated inverse:

$$F_\tau = 1 - e^{-\tau w_R}$$

$$F^{-1}(\tau) = \frac{-1}{w_R} \log(1 - u)$$

Roulette selection

How to sample a reaction based on propensities

Compute Propensities

Compute associated cumulative sums

Sample *u* from a uniform distribution between 0 and the sum of all propensities

Go through cumulative propensities; stop when *u* < *c*ᵢ

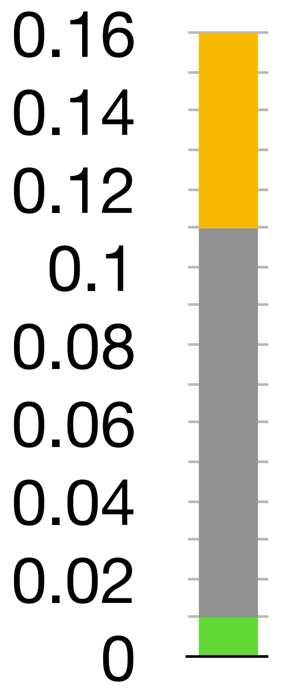
$$w_1 = 0 \quad \rightarrow c_1 = 0$$

$$w_2 = 0.01 \quad \rightarrow c_2 = 0.01$$

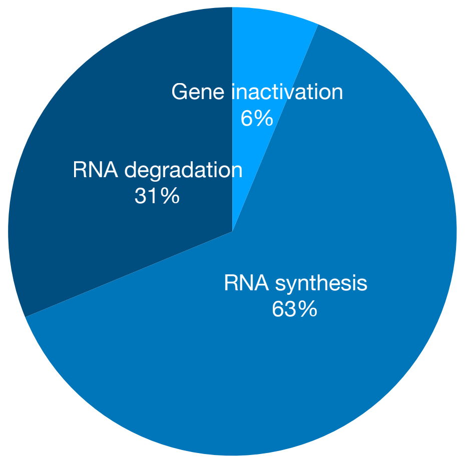
$$w_3 = 0.1 \quad \rightarrow c_3 = 0.11$$

$$w_4 = 0.05 \quad \rightarrow c_4 = 0.16$$

$$u \sim Unif(0, \sum_{i \in R} w_i)$$



$$u < c_1?$$
$$u < c_2?$$
$$\vdots$$



With r_1 and r_2 two random numbers from the unit-interval uniform distribution, take

$$\tau = (1/a_0) \ln (1/r_1) \tag{21a}$$

and take μ to be that integer for which

$$\sum_{\nu=1}^{\mu-1} a_\nu < r_2 a_0 \leqslant \sum_{\nu=1}^{\mu} a_\nu \tag{21b}$$

The generating procedure (21) is easy to code in Fortran.

```
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import clear_output
import random
```

Week 10 Section:

Gillespie Algorithm and master equations

Things to remember:

- Master equations can be defined in terms of stepped increments and updates.
- The Gillespie SSA algorithm allows us to sample probability distributions described by master equations.

The Gillespie SSA algorithm:

The propensities are nothing else than the transition probabilities from one state to the next. The propensity for a given transition (reaction) r is denoted as w_r

Let's write a function implementing the gillespie algorithm for a similar problem described in class we can write each change of state - the copy number of the mRNA and the availability of the gene - and their respective propensities:

r	reaction	propensity	η(R)	Description
1	$Gene(I) \xrightarrow{k_b} Gene(A)$	k_b	0	Gene activation
2	$Gene(A) \xrightarrow{k_u} Gene(I)$	k_u	0	Gene inactivation
3	$Gene(A) \xrightarrow{k_1} Gene(A) + RNA$	k_1	+1	RNA synthesis
4	$RNA \xrightarrow{k_2} \emptyset$	$k_2 R$	-1	RNA degradation

The events that we outlined above are going to be rare, discrete and independent. Each one of them is the occurrence of a Poisson process and we'll go along the lines of the following logic, but before a couple of things to keep in mind:

States changes in our system at a Δt (which we know is drawn from an exponential distribution with mean W_R) any of our reactions can happen, but the probability that reaction r happens is going to be proportional to w_r. Reactions with higher propensities are more likely to happen.
To choose which reaction i is going to happen out of the possible ones we can reduce the problem to sampling a random number in the interval from 0 to 1, where the drawing probability of each state is:

$$\frac{w_i}{\sum_r w_r} = \frac{w_i}{W_R}$$

The reason that the *Dwell time* is sampled from an exponential distribution with mean W_R is this:

Imagine we had just one Poisson distributed set of events, we know that the waiting time between events is exponentially distributed.

Another way of looking at it is the probability that the elapsed time t is greater than Δt:

$$P(t > \Delta t \mid w_1) = \int_{\Delta t}^{\infty} \mathrm{d}t \; P(t \mid w_1) = \mathrm{e}^{-w_1 \Delta t}$$

Imagine now that you have multiple poisson events that can happen and similarly, the probability that no event has happened is:

$$P(t_1 > \Delta t, t_2 > \Delta t, \ldots) = P(t_1 > \Delta t) P(t_2 > \Delta t) \cdots = \prod_r \mathrm{e}^{-w_r \Delta t} = \mathrm{e}^{-\Delta t \sum_r w_r} = \mathrm{e}^{-\Delta t W_R}$$

which would be equivalent to the probability of a single poisson process with $w = \sum_r w_r$ the probability that it does happen in the Δt is exponentially distributed with mean $\frac{1}{\sum_r w_r}$:

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

Where τ is our dwell time.

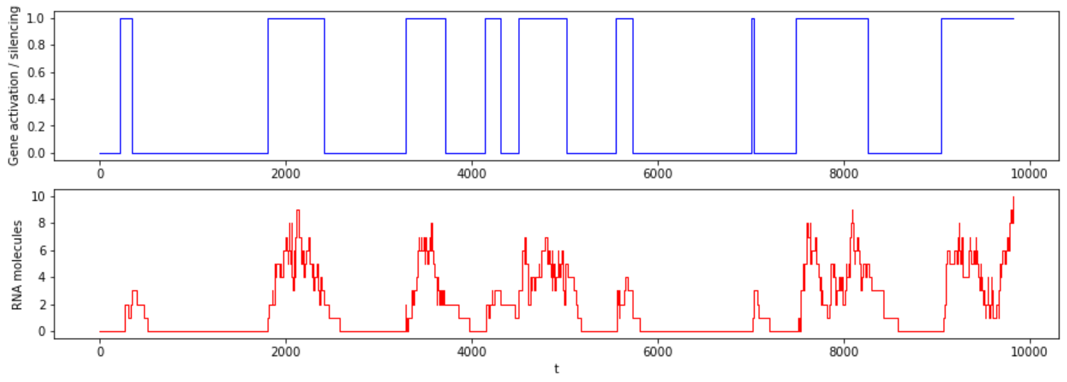
Now the logic that we are following:

1. start the algorithm in some state:
   - Gene: Active or Inactive.
   - mRNAs: Any number of them.
2. Calculate all the propensities, they could be a function of the state of the system -something to watch out for- they need to be computed at every step.
3. Sample a dwell time
4. Sample a transition
5. Increment the time by τ
6. re-write the states in our system

```python
def reaction(kb, ku, k1, k2, T, dynamic_plotting = False):
    # initialize our states
    ga,r,t = [0],[0],[0]

    propensities_history = []

    while (t[-1] < T):

        # calculate the propensities
        # some of which rely on our current state
        # (whether the gene is active, the number of mRNAs, etc.)

        propensities = np.array([kb * (1-ga[-1]), ku * ga[-1], k1 * ga[-1], k2 * r[-1]])
        propensities_history.append(propensities)
        # sample a dwell time
        tau = (-1/sum(propensities)) * np.log(np.random.random())

        # sample a reaction
        gillespie_r = random.random()
        # Update our states
        if gillespie_r ⩽ np.cumsum(propensities/sum(propensities))[0]: #kb
            ga+= [1] # equivalent to ga.append(1)
            r += [r[-1]] #  equivalent to r.append(r[-1])
        elif gillespie_r <= np.cumsum(propensities/sum(propensities))[1]:#ku
            ga+= [0]

            r += [r[-1]]
        elif gillespie_r <= np.cumsum(propensities/sum(propensities))[2]: #k1
            r += [r[-1] + 1]

            ga+= [ga[-1]]
        elif gillespie_r <= np.cumsum(propensities/sum(propensities))[3]: #k2
            r += [r[-1] - 1]

            ga+= [ga[-1]]

        # increment the time by tau
        t += [t[-1] + tau]

        if dynamic_plotting == True:
            if len(t) % 100 == 0:
                clear_output(wait=True)
                fig,ax = plt.subplots(ncols= 1,nrows =2)
                fig.set_figwidth(15)
                fig.set_figheight(5)

                ax[1].step(t, r , lw = 1,c = 'r', label = 'rna', where='post')
                ax[0].step(t, ga , lw = 1,c = 'b', label = 'Gene', where='post')

                ax[1].set_xlabel('t')
                ax[0].set_ylabel('Gene activation / silencing')
                ax[1].set_ylabel('RNA molecules')

                #ax[1].step(t+[T], [0]+r , lw = 1,c = 'r', label = 'rna')
                #ax[0].step(t+[T], [0]+ga , lw = 1,c = 'b', label = 'Gene')
                plt.show();
    return t,ga,r, propensities_history
```

r	reaction	propensity	η(R)	Description
1	$Gene(I) \xrightarrow{k_b} Gene(A)$	k_b	0	Gene activation
2	$Gene(A) \xrightarrow{k_u} Gene(I)$	k_u	0	Gene inactivation
3	$Gene(A) \xrightarrow{k_1} Gene(A) + RNA$	k_1	+1	RNA synthesis
4	$RNA \xrightarrow{k_2} \emptyset$	$k_2 R$	-1	RNA degradation

```
t,ga,r, propensities_history = reaction(kb = 0.002, ku = 0.001, k1 = 0.05, k2 = 0.01,T = 10000, dynamic_plotting=True)
```



For the homework

r	reaction	propensity	η(R)	η(P)	Description
1	$Gene(I) \xrightarrow{k_b} Gene(A)$	k_b	0	0	Gene activation
2	$Gene(A) \xrightarrow{k_u} Gene(I)$	k_u	0	0	Gene inactivation
3	$Gene(A) \xrightarrow{k_1} Gene(A) + RNA$	k_1	+1	0	RNA synthesis
4	$RNA \xrightarrow{k_2} \emptyset$	$k_2 R$	-1	0	RNA degradation
5	$RNA \xrightarrow{k_3} RNA + Protein$	$k_3 R$	0	+1	Protein synthesis
6	$Protein \xrightarrow{k_4} \emptyset$	$k_4 P$	0	-1	Protein degradation

```
def reaction_rp_model(kb, ku, k1, k2, k3, k4, T, dynamic_plotting = False):
    # initialize our states
    ga, r, p, t = [0],[0],[0],[0]
    propensities = np.array([kb * (1-ga[-1]), ku * ga[-1], k1 * ga[-1], k2 * r[-1], None, None, None]) # replace None

    propensities_history.append(propensities)
    # sample a dwell time
    tau = (-1/sum(propensities)) * np.log(np.random.random())

    # sample a reaction
    gillespie_r = random.random()
    # Update our states
    ###
    ### ???
    ### ???
    ###
    # increment the time by tau
    t += [t[-1] + tau]
    if dynamic_plotting == True:
        if len(t) % 100 == 0:
            clear_output(wait=True)
            fig,ax = plt.subplots(ncols= 1,nrows =2)
            fig.set_figwidth(15)
            fig.set_figheight(5)

            ax[0].step(t, ga , lw = 1,c = 'b', where='post')
            ax[1].step(t, r , lw = 1,c = 'r', where='post')
            ax[2].step(t, p , lw = 1,c = 'g', where='post')
            plt.show();
    return t,ga,r,p, propensities_history
```