Simple models of diffusion - 2-state model

Exercise Report: Markov chain

(Whole code at the end of the report)

Introduction

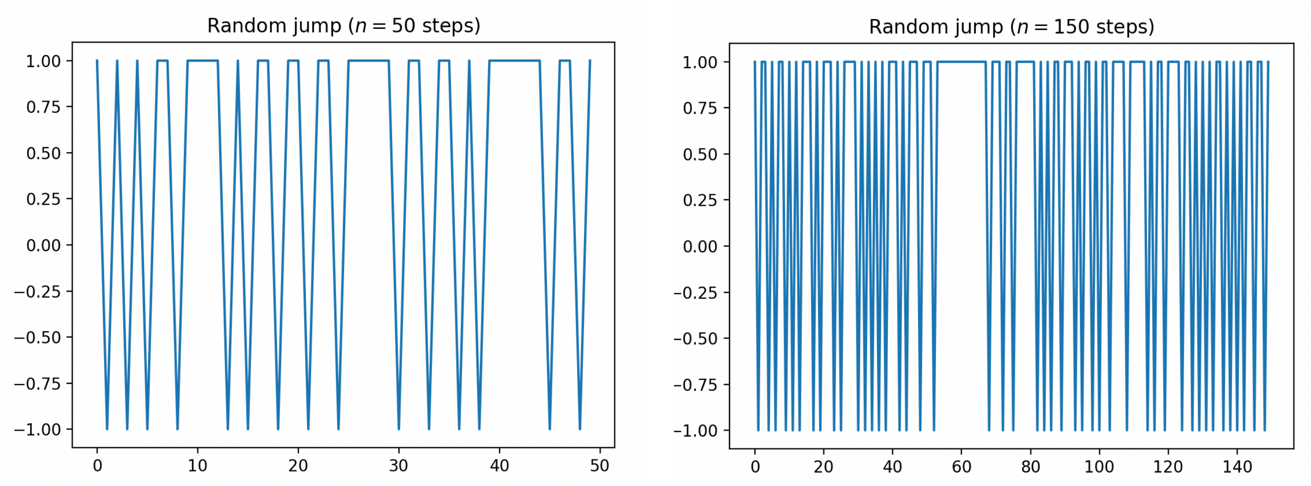
We have flee that jump from one dog to another. This flee have probability for jump from state 1 to -1 equals to 1/2 ($\alpha = 1/2$), but from -1 to 1 probability is equal to 1 ($\beta = 1$). We analyze state distribution using Markov chain. Our expected value is $\pi_1 = 1/3$ and $\pi_2 = 2/3$.

We can simulate this situation by provided code:

```
import numpy as np
import matplotlib.pyplot as plt
import random

def random_jump(N, K):
    for k in range(K):
        s = np.zeros(N)
        pi_vales = np.zeros(N)
        s[0] = 1
        T = [[[1/2], [1/2]],
                [[1], [0]]]
        count = 0

        π1 = np.zeros(N)
        π1[0] = 0

        π2 = np.zeros(N)
        π2[0] = 1

        for i in range(1, N):
            u = random.random()

            if s[i-1] == 1:
                if u >= T[0][1][0]:
                    s[i] = 1
                    count += 1
                else:
                    s[i] = -1
            else:
                s[i] = 1

            if count/i >= 1:
                print(count, i)

            π1[i] = (count/i)
            π2[i] = (1-count/i)

            pi_vales[i] = 1 - π2[i]

        π1_values = np.mean(pi_vales)
        π2_values = 1 - π1_values

    return π1, π2, s, π1_values, π2_values
```

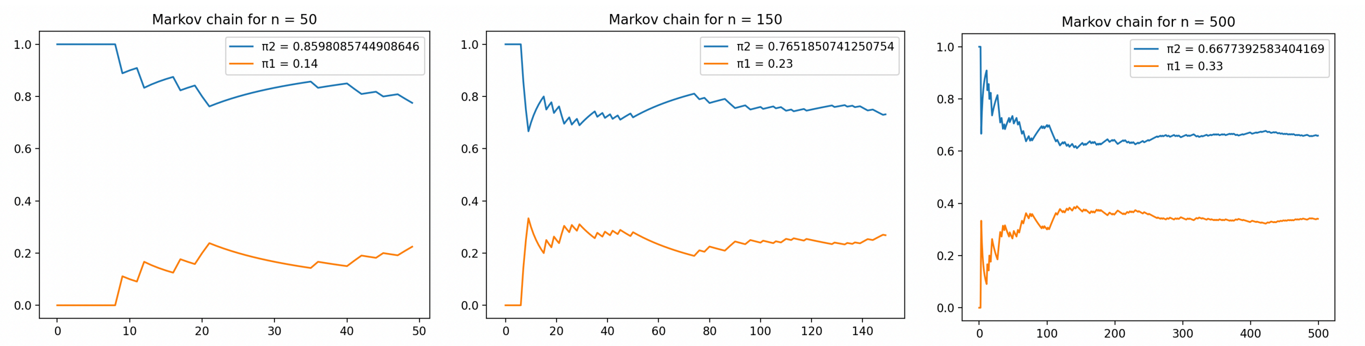
(Python 3.11.7)

Those are the result:



1) Random jump for n = 50, 150 for $\alpha = 1/2$ and $\beta = 1$

Now we can plot Markov chain for out simulation:



2) Markov chain for n=50, 150, 500

We can see that with increasing number of random jump, our simulated π_1 and π_2 getting closer to expected values ($\pi_1 = 1/3$ and $\pi_2 = 2/3$).

```(WHOLE CODE)

```
import numpy as np
import matplotlib.pyplot as plt
import random

def random_jump(N, K):
    for k in range(K):
        s = np.zeros(N)
        pi_vales = np.zeros(N)
        s[0] = 1
        T = [[[1/2], [1/2]],
                [[1], [0]]]
        count = 0

        π1 = np.zeros(N)
        π1[0] = 0

        π2 = np.zeros(N)
        π2[0] = 1

        for i in range(1, N):
            u = random.random()

            if s[i-1] == 1:
                if u >= T[0][1][0]:
                    s[i] = 1
                    count += 1
                else:
                    s[i] = -1
            else:
                s[i] = 1

            if count/i >= 1:
                print(count, i)

            π1[i] = (count/i)
            π2[i] = (1-count/i)

            pi_vales[i] = 1 - π2[i]

        π1_values = np.mean(pi_vales)
        π2_values = 1 - π1_values

    return π1, π2, s, π1_values, π2_values

def main():

    N = 150 # Number of steps in markov chain
    fig, ax = plt.subplots(2, figsize=(12, 10))

    π1, π2, s, π1_values, π2_values = random_jump(N, 1)

    print(π1_values, π2_values)

    ax[0].plot(s)
    ax[0].set_title("Random jump ($n = {}$ steps)".format(N))

    ax[1].plot(π2, label='π2 = {}'.format(π2_values))
    ax[1].plot(π1, label='π1 = {}'.format(round(π1_values, 2)))
    ax[1].set_title("Markov chain")
    ax[1].legend()

    plt.show()

if __name__ == "__main__":
    main()
```

```Python 3.11.7)