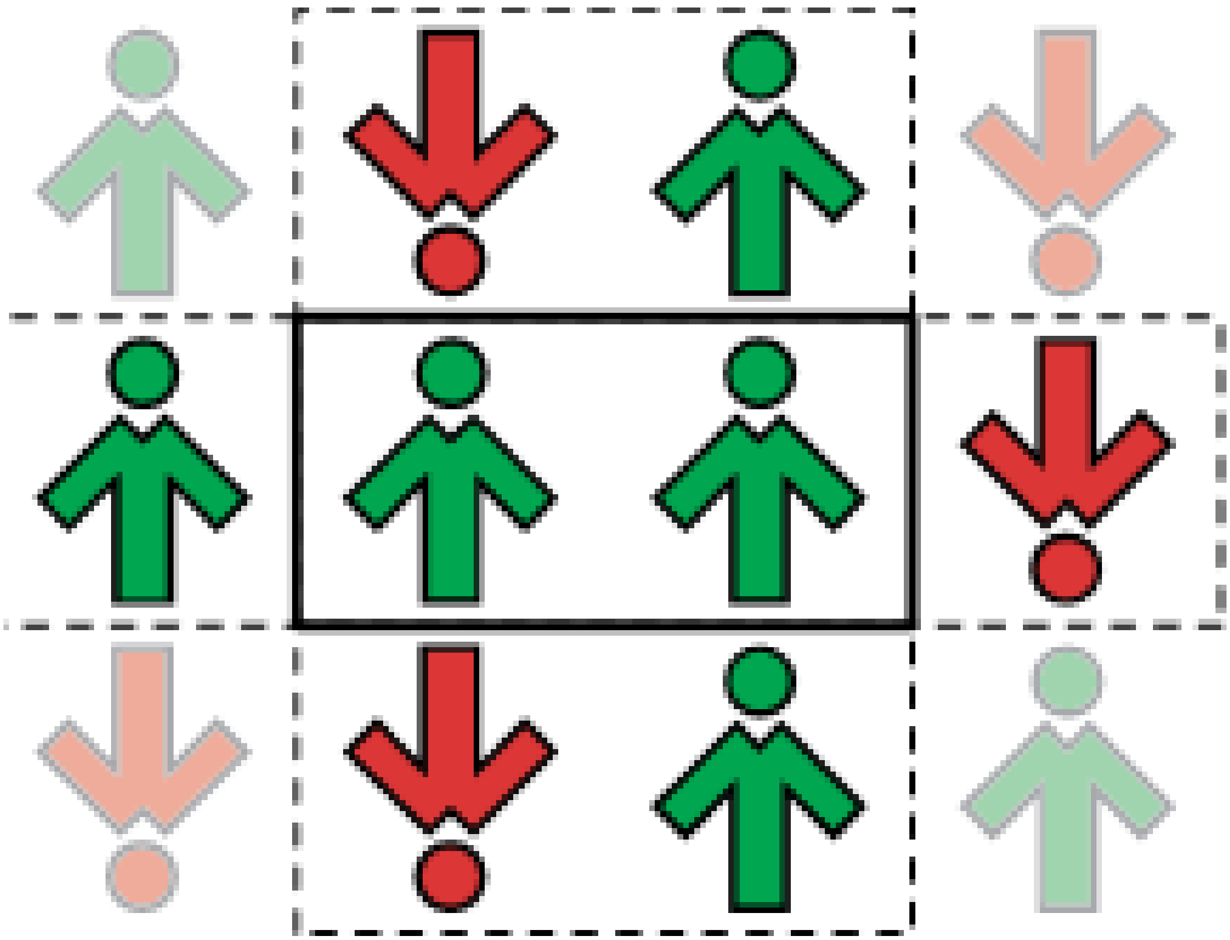
Business Intelligence Workspace Project 3

Opinion Dynamics – Prescriptive Analytics

Sznajd model with advertising



Submitted by: *Hammad Ullah*

Submitted to: *Piotr Nyczka*

Date: *18-June-2022*

Background:

The Sznajd model is essentially a socio-physics models that comes under the realm of Agent Based Simulations that helps us understand and model the complexities of the real world. The Sznajd model helps us comprehend opinion dynamics by implementing what's called a social validation phenomenon. Opinion dynamics is basically a set of rules a user sets to model how social conformity can bring about change in opinion amongst the nearest neighbors of a person(s). However, in this report we shall extend the model to include the effect of an advertisement campaign as well. Thus, the opinion dynamics are going to be based on social conformity (social influence) as well as advertisement (external influence).

Aim and Tools:

The aim of this report is to analyze the effect of social conformity and advertisement in an environment where people hold two different opinions. In order to quantify and compare our results, we shall use different set of parameters to analyze the effect of both types of influences together. Moreover, we shall analyze the time steps taken to influence 75% of the population to have a certain opinion. Since more time steps would mean longer advertisement campaign, the total cost of the advertisement campaign is also formulated and will be calculated in order to analyze the results and choose the best intensity of the advertisement campaign and the length of the campaign that we are willing to pay for.

We start with initial condition of a green square droplet of the size **d*d**, in the red sea of size **L*L** as seen in the Figure 1 below:

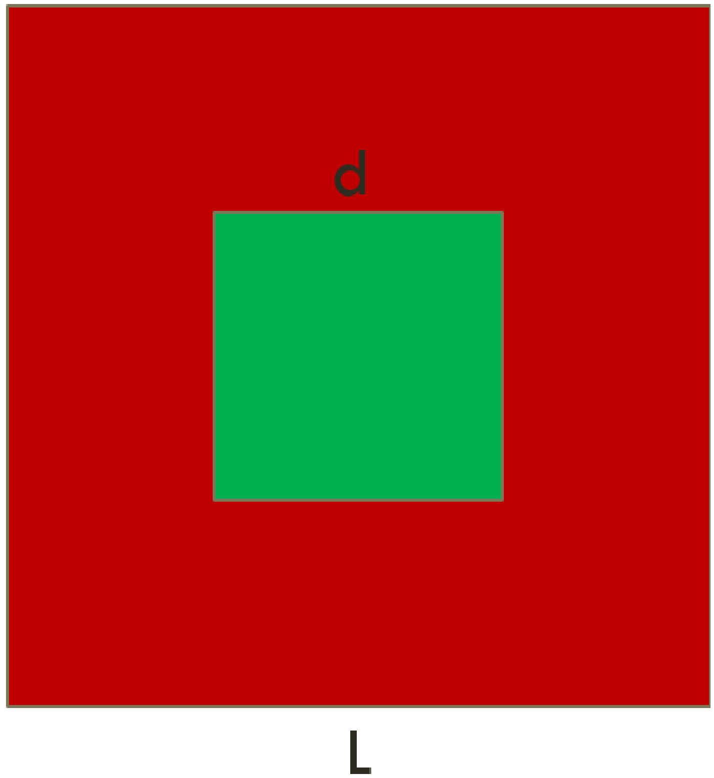


Figure 1 Initial Sea with droplet

Then, we implement the two types of influence. Before we go into the implementation part, lets study how each influence will work.

<u>Conformity (Social Influence):</u>

Agents inside solid frame influence their neighbors (von Neuman) in dotted frame, but only if there is an agreement inside soild frame as seen in the Figure 2 below. Faded agents do not take part in the reaction.

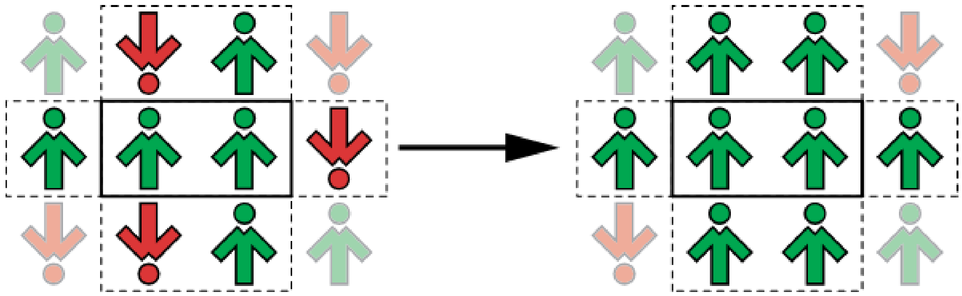


Figure 2 Conformity (Social Influence)

<u>Advertisement (External Influence):</u>

Sometimes (with probability p per step) an agent is switched upwards, as a result of advertisement as seen in the Figure 3 below. A point to be noted here is that the advertisment is only one directional for our case.

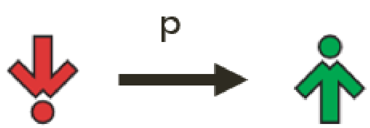


Figure 3 Advertisement (External Influence)

After starting the initial followers' conditions, we toss a dice of random continuous numbers between **0** and **1**, and if the probability is p or less, we chose a random cell from the **L*L** matrix and change its opinion to that of the cells in the initial droplet. If the opinion is already in agreement to that, we leave it as it is. On the other hand, if the probability of the number we get is greater than p, then we chose two adjacent cells, they can be horizontal or vertical, and apply the social conformity rules. In order to tackle with the issue to neighbors of the cells chosen at the boundary, we use periodic boundary conditions, in order words we implement a torus world topology as shown in the Figure 4 below.

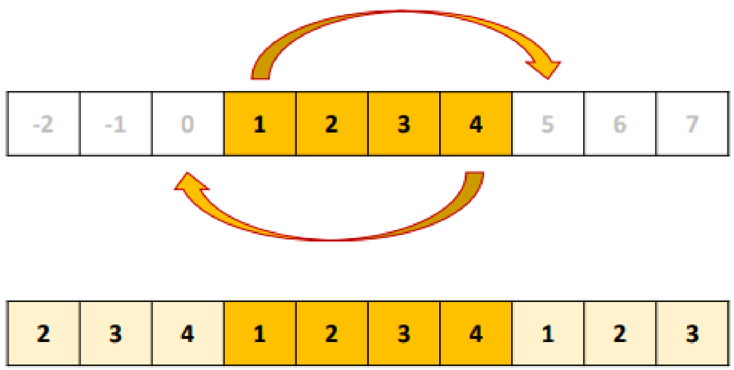


Figure 4 Periodic Boundary Conditions

We keep repeating these steps until a certain percentage of the agents (75 %) hold the same opinion as of those agents in the initial droplet or till 100 million steps i.e., whichever comes first. Next, we calculate the price using a formula that integrates the number of steps taken until success and the intensity of the advertisement campaign. We replicate the same loop 10 times in order to make sure our results are precise and take the average of the price.

This is repeated for all the different combinations of p and d. The rest of the parameters remain constant. The different parameters are listed below:

- **p** – advertisement intensity = [0.08, 0.09, 0.10, 0.11, 0.12, 0.13, 0.14, 0.15]
- **d** – droplet size (**d*d**) = [0, 6, 10, 14]
- **Maximum Number of Steps** = 100 million
- **Number of replications** = 10
- **Destination** = 75%

- **L** – sea size (**L*L**) = 20
- **X** – Price Formula – **X = f(p)*t + g(d)** – where $f(p) = 600*p^6$ & $g(d) = d^2$

Our task is to design such a campaign that ends with convincing 75% of the agents, keeping in mind, the minimization of the cost which is a function of the parameters p, d and of course time.

We shall use **MATLAB** to implement this task.

Code and Methodology:

```
clc;
clear;
```

- This initial piece of code is just to make sure that command window as well as variable workspace is cleared, every time we run the code.

```
L = 20;
T = 100000000;
allignment = ["horizontal" "vertical"];
prob = [0.08:0.01:0.15];
d = [0 6 10 14];
```

- Above, we have defined the different parameters. L is the total matrix size; T is the maximum number of time steps to stop the loops if we are not able to reach the destination point which is 75%.
- Since, for conformity we can choose a 1 by 2 horizontal block or a 2 by 1 vertical block, I created a vector of strings for that as well.
- The different intensities of advertisement and the different initial droplet size is also defined as **prob** and **d** respectively.

```
% Time loop
row = 0;
tab = [];
for z = 1:4
    for k = 1:8
        for rep = 1:10
```

- We begin the loops in the above code. **z** goes from 1 to 4 through the 4 values of droplet size, **k** goes from 1 to 8 through all the 8 different values of advertisement intensities and **rep** goes from 1 to 10 for all the 10 different replications of the loops.

```
if z == 1
    S = zeros(L,L) - 1;
    figure(1)
    imshow(S,'InitialMagnification','fit')
    title(['Time step: 0 out of ' num2str(T)])
    pause(1)
elseif z == 2
    S = zeros(L,L) - 1;
    S(8:13,8:13) = 1;
    figure(1)
    imshow(S,'InitialMagnification','fit')
    title(['Time step: 0 out of ' num2str(T)])
    pause(1)
elseif z == 3
    S = zeros(L,L) - 1;
    S(6:15,6:15) = 1;
    figure(1)
    imshow(S,'InitialMagnification','fit')
    title(['Time step: 0 out of ' num2str(T)])
    pause(1)
elseif z == 4
    S = zeros(L,L) - 1;
    S(4:17,4:17) = 1;
    figure(1)
    imshow(S,'InitialMagnification','fit')
    title(['Time step: 0 out of ' num2str(T)])
    pause(1)
end
```

- After initializing the loops, we must make our initial droplet and matrix which depend on the value of **d** only as **L** remains constant. This for every different value of **d**, the matrix and droplet has been defined. The matrix without the droplet is given the values of **-1** whereas the cells falling into droplet space is given the value of **1**
- Then we plot the graph using **imshow** function which plots the -1 values as **black** and 1 as **white**. Thus, the matrix will be a black window with the droplet being white in the center, if d is greater than zero.

```
    for t= 1:T
        if (sum(S(:) == 1)) >= 300
            success = 1;
            fp = 600*(prob(k))^6;
            gd = d(z)^2;
            X = fp*t + gd;
            tab(row,1:5) = [prob(k),d(z),t-1,success, X]
            break
        end
        % Randomly select a cell (an agent), say S(i,j) with i,j in {1,…,N}
        i = floor(rand*L)+1;
        j = floor(rand*L)+1;
        p = rand();
        allign = randsample(allignment, 1);
```

- After the matrix has been made, we take random cell indices as **i** and **j**. We take a random probability value between **0** and **1**. Lastly, we choose a random alignment, horizontal or vertical.
- Before going through the timestep, it also checks if the destination condition (**75%**) is satisfied or not. If it is, it breaks out of the loop. Before it breaks out, the price is calculated using the price function formula. The parameters and value of X is stored into a row of a table generated earlier.

```
        if p >= prob(k)
            if allign == 'horizontal'
                if S(i,j)*S(i,mod(j+1-1,L)+1)==1
                    % Modify agents above the panel
                    S(mod(i-1-1,L)+1,j)=S(i,j);
                    S(mod(i-1-1,L)+1,mod(j+1-1,L)+1)=S(i,j);
                    % Modify agent to the left of the panel
                    S(i,mod(j-1-1,L)+1)=S(i,j);
                    % Modify agent to the right of the panel
                    S(i,mod(j+2-1,L)+1)=S(i,j);
                    % Modify agents below the panel
                    S(mod(i+1-1,L)+1,j)=S(i,j);
                    S(mod(i+1-1,L)+1,mod(j+1-1,L)+1)=S(i,j);
                end
                figure(1)
                % Plot selected panel in gray
                S1_old = S(i,j); S2_old = S(i,mod(j+1-1,L)+1);
                S(i,j) = 0; S(i,mod(j+1-1,L)+1) = 0;
                % Plot system state: 0 - black, 1 - white, (0,1) - gray
                imshow((S+1)/2,'InitialMagnification','fit')
                    title(['Time step: ' num2str(t) ' out of ' num2str(T) ...
                    ', selected agent: (' num2str(i) ',' num2str(j) ')'])
                %pause(1)
                % Set panel state to original values
                S(i,j) = S1_old; S(i,mod(j+1-1,L)+1) = S2_old;
```

- Here, we first check if the random probability generated is greater than advertisement intensity, then we apply the conformity rules where we initially check if both the cells chosen are similar to each other. Since, the white cells are 1 and black cells are 1, we will get a product equal to one only if we chose both white or both black cells. The excessive use of mod function is necessary in order to implement periodic boundary condition that depicts a torus world topology. After that, we plot the graph.

```
else
    if S(i,j)*S(mod(i+1-1,L)+1,j)==1
        % Modify agents above the panel
        S(mod(i-1-1,L)+1,j)=S(i,j);
        %S(mod(i-1-1,L)+1,mod(j+1-1,L)+1)=S(i,j);
        % Modify agent to the left of the panel
        S(i,mod(j-1-1,L)+1)=S(i,j);
        S(mod(i+1-1,L)+1,mod(j-1-1,L)+1)=S(i,j);
        % Modify agent to the right of the panel
        S(i,mod(j+1-1,L)+1)=S(i,j);
        S(mod(i+1-1,L)+1,mod(j+1-1,L)+1)=S(i,j);
        % Modify agents below the panel
        S(mod(i+2-1,L)+1,j)=S(i,j);
    end
    figure(1)
    % Plot selected panel in gray
    S1_old = S(i,j); S2_old = S(mod(i+1-1,L)+1,j);
    S(i,j) = 0; S(mod(i+1-1,L)+1,j) = 0;
    % Plot system state: 0 - black, 1 - white, (0,1) - gray
    imshow((S+1)/2,'InitialMagnification','fit')
        title(['Time step: ' num2str(t) ' out of ' num2str(T) ...
        ', selected agent: (' num2str(i) ',' num2str(j) ')'])
    %pause(1)
    % Set panel state to original values
    S(i,j) = S1_old; S(mod(i+1-1,L)+1,j) = S2_old;
end
```

➢ Here, the conformity rules are applied again for the case where the alignment is vertical.

```
    end
else
    if S(i,j)==-1
        S(i,j) = 1;
        figure(1)
        % Plot selected panel in gray
        S1_old = S(i,j);
        S(i,j) = 0;
        % Plot system state: 0 - black, 1 - white, (0,1) - gray
        imshow((S+1)/2,'InitialMagnification','fit')
            title(['Time step: ' num2str(t) ' out of ' num2str(T) ...
            ', selected agent: (' num2str(i) ',' num2str(j) ')'])
        %pause(1)
        % Set panel state to original values
        S(i,j) = S1_old;
    end
end
```

➢ The above code is if the random probability chosen is less than the advertisement intensity. In which case, we apply the advertisement rules where we take a random cell and change it from black to white in our case.

```
            if t == T
                success = 0;
                fp = 600*(prob(k))^6;
                gd = d(z)^2;
                X = fp*t + gd;
                tab(row,1:5) = [prob(k),d(z),t,success,X]
                break
            end
        end
    end
end
end
```

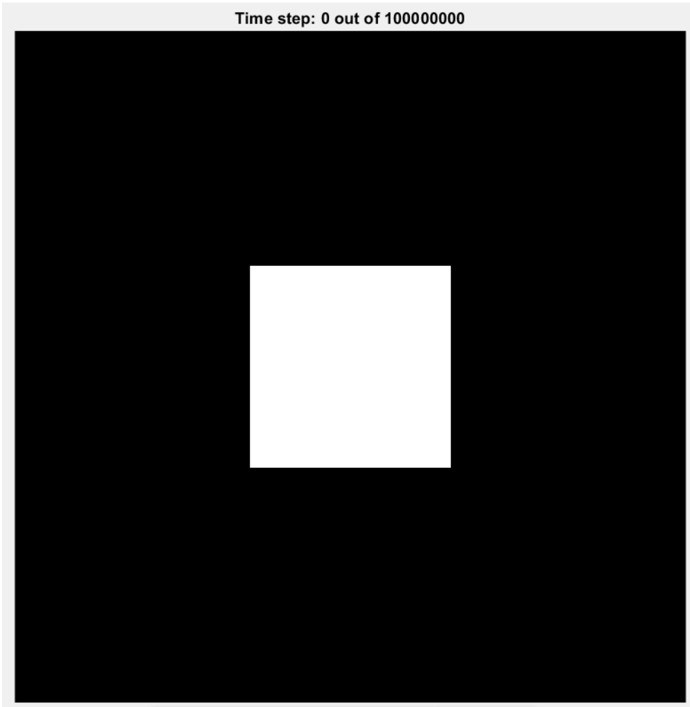
- Here we use another break statement when the number of time steps has reach maximum value i.e., 100 million. And we add the parameters and the X value into a row of the table.

```
column_names = {'p','d','t','success','X'};
TABLE = array2table(tab, 'VariableNames', column_names);
writetable(TABLE,'results.csv');
```

- Lastly, we label our table with column names and save it as a csv file.

Results and Analysis:

An example of the matrix with droplet generated by our code can be seen below:



The droplet can be seen in white which means that our advertisement, which is one directional, will be turning the black agents to white. However, the conformity rules apply to both agents.

A preview of the results generated from the simulation can be seen below:

	A	B	C	D	E
1	p	d	t	success	X
2	0.08	0	3120748	1	490.8513755
3	0.08	0	1744162	1	274.3331193
4	0.08	0	2433930	1	382.8242448
5	0.08	0	4593688	1	722.5248055
6	0.08	0	2217572	1	348.7940739
7	0.08	0	2421924	1	380.9358643
8	0.08	0	1526659	1	240.1228554
9	0.08	0	7019284	1	1104.038068
10	0.08	0	606297	1	95.36242975
11	0.08	0	19424	1	3.05528832

However, it is difficult to analyze the results this way, therefore, we shall transform this first into the results of the averages of the 10 simulations and then put them into a grid.

12	0.09	0	17796	1	5.674833286
13	0.09	0	179012	1	57.08090864
14	0.09	0	905510	1	288.7354028
15	0.09	0	1835885	1	585.399055
16	0.09	0	57494	1	18.33312018
17	0.09	0	244316	1	77.90404248
18	0.09	0	857823	1	273.5297066
19	0.09	0	3940056	1	1256.344699
20	0.09	0	793558	1	253.0378731
21	0.09	0	42840	1	13.66047833
22	0.1	0	305842	1	183.5058
23	0.1	0	62477	1	37.4868
24	0.1	0	517232	1	310.3398
25	0.1	0	49088	1	29.4534
26	0.1	0	23805	1	14.2836

Below, we can see the results of the 10 simulations as an average of price (Xavg) and number of time steps (t) for every combination of advertisement intensity (p) and droplet size (d). Here, it is easier to see that as the droplet size increases, the number of time steps to reach destination decreases and the average price for the campaign also decreases. This is expected as a higher droplet size means high number of initial followers and thus higher chance of social conformity and less intense advertisement campaign is needed to reach the destination. Moreover, as the advertisement intensity increases, the number of time steps to reach destination decreases and the average price for the campaign also decreases. This was also expected as a more intense advertisement campaign means more people are probable to be convinced in a time step and thus less time steps are needed on average to reach the destination.

p	**d**	**t**	**X (avg)**
0.08	0	2570368.8	404.2842125
0.09	0	887429	282.970012
0.1	0	264315.2	158.58972
0.11	0	225596.4	239.7957333

0.12	0	174928.3	313.4016546
0.13	0	108986	315.6356595
0.14	0	47381.2	214.0595884
0.15	0	23756.8	162.3697144
0.08	6	2486778.7	427.1366266
0.09	6	1126999	395.3604042
0.10	6	413611.3	284.16738
0.11	6	232508.6	283.1429637
0.12	6	53148.6	131.2223131
0.13	6	54270.8	193.1757676
0.14	6	28138.9	163.1282341
0.15	6	19833.2	171.5543606
0.08	10	1708950.1	368.7947663
0.09	10	163219.9	152.045367
0.10	10	111787.3	167.07298
0.11	10	69160.8	173.5146085
0.12	10	57247.9	202.5665797
0.13	10	23945.4	169.3508194
0.14	10	11108.7	150.1905317
0.15	10	7069.3	148.3210816
0.08	14	228323.6	231.9123544
0.09	14	12545	200.0004753
0.10	14	81560.8	244.93708
0.11	14	4893.8	201.2028621
0.12	14	3386.4	202.0688333
0.13	14	3407.3	205.8707279
0.14	14	4599.3	216.7828747
0.15	14	2489.6	213.0216944

In order to even better visualize and understand our results, we shall display them in the form of a grid where on y-axis, we shall have the droplet size values, whereas on the x-axis, we shall have the p values. On the grid itself, we shall display the average price value that resulted from the function after 10 repetitions.

The grid can be seen below:

X(avg)

d \ p	0.08	0.09	0.1	0.11	0.12	0.13	0.14	0.15
14	232	200	245	201	202	206	216.8	213
10	369	152	167	174	203	169	150.2	148
6	427	395	284	283	131	193	163.1	172
0	404	283	159	240	313	316	214.1	162

We can see that the greater the initial number of followers, i.e., the greater the droplet size, the lesser intense campaign we need to carry out plus the advertisement campaign costs less to reach destination.

We can make a similar grid for d versus p for number of time steps as well. We can see this grid below too:

Steps to reach destination

d \ p	0.08	0.09	0.1	0.11	0.12	0.13	0.14	0.15
14	228324	12545	81561	4893.8	3386.4	3407.3	4599.3	2489.6
10	1708950	163220	111787	69161	57248	23945	11109	7069.3
6	2486779	1126999	413611	232509	53149	54271	28139	19833.2
0	2570369	887429	264315	225596	174928	108986	47381	23756.8

Above can see that the greater the initial number of followers, i.e., the greater the droplet size, the lesser is the time the campaign needs to be run for to reach destination. Moreover, the greater the intensity of the advertisement campaign, the lesser is the time needed for the campaign to be ran for.

Conclusion

Our conclusion to this study might help a company to decide how intense they want their advertisement campaign to be and how much time it is going to take in order to achieve their goals. Firstly, as we can see from the results above, the cost of the advertisement campaign depends on two main factors, the intensity of the advertisement campaign, and the initial droplet size which represents the initial number of users of that product/service. To explain further, the intense the advertisement campaign is, the more is the cost, and the lower the initial number of followers/users there are, the higher the cost is to reach the destination. The latter is because, the lower the initial number of users, the more time the advertisement campaign needs to be run for. Since, the cost of the campaign depends on the time as well, the more time the campaign needs to be run for, the more is the cost of the advertisement campaign. Such simulations and their analysis is necessary for companies to take informed decisions regarding marketing their product as well reducing the cost of marketing itself.